Example
Tech Documentation

Secure Payment APIs v5.51

Released: **2011-01-10** | Sprint: 5.51
Security classification: **PUBLIC**

Purpose

The purpose of the Example Payment APIs collection is to describe how to integrate with Example's standard RESTful API from external servers such as Portal, Membership, Game, and so on.

Example Secure Payment APIs Security: Encrypted Data

The Example Secure Payment APIs protect all HTTPS Request and Response data according to PCI and PII standards with the AES-GCM encryption *(https://tools.ietf.org/html/rfc5288)* (Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) as a Transport Layer Security (TLS) authenticated encryption operation). Please see the Protocol Rules "AES-GCM encrypted request and response data" section for these and other Example Secure Payment API security-related details.

Changelog / Version History

Version 5.51 - 2011/01/10
This release includes the following updates:

- **Updated**
  - **Payout Transaction API**
    - Added JSON property `"region"` to the response.
  - **Refund Transaction API**
    - Added JSON property `"region"` to the response.
  - **Test Card Information**
    - Updated the E-Com Processing Test Card Information.

Payment API endpoints overview

Payment API endpoint url schemes:

- Example Development Server: **https://pl.dev.Example.net**
- Example QA Server: **https://pl.qa.Example.net**
- Example Prod Server: **https://plpp.qa.Example.net**

Account APIs

See the Account APIs page for further data.

DELETE **CancelPaymentInfoByOrderId**

URL: /account/{uuid}/orders/{orderId}
Description: Cancels PaymentInfo via a mobile client OrderID.

POST **CreatePaymentInfo**

URL: /account/{uuid}/paymentinfo

Description: Create a user’s payment information.

PUT **UpdatePaymentInfo**

URL: /account/{uuid}/paymentinfo/{paymentInfoId}
Description: Update user’s payment information.

DELETE **DeletePaymentInfo**

URL: /account/{uuid}/paymentinfo/{paymentInfoID}
Description: Delete user’s payment information.

GET **GetPaymentInfo**

URL: /account/{uuid}/paymentinfo
Description: Get user's payment information.

GET **GetPaymentinfoByPaymentInfoID**

URL: /account/{uuid}/paymentinfo/{paymentInfoId}
Description: Get Payment Information by paymentInfoID.

GET **GetCardExpiries**

URL: /account/{uuid}/paymentinfo/expiries
Description: Gets a list of a user’s expired credit cards.

POST **ReVerifyPaymentInfo**

URL: /account/{uuid}/paymentinfo/{paymentInfoID}/verification
Description: Reverify payment information by uuid and paymentInfoID.

Billing APIs

See the Billing APIs page for further data.

DELETE **CancelTransactionByOrderId**

URL: /billing/{uuid}/orders/{orderId}
Description: Cancels a transaction by mobile client orderID.

POST **CheckTransaction**

URL: /billing/{uuid}/transactions/check
Description: Checks a transaction before deposit. (3DSecure)

POST **CreateTransactionWithNewPaymentInfo**

URL: /billing/{uuid}/transactions
Description: Do a payment process with new payment information.

POST **CreateTransactionWithPaymentInfoID**

URL: /billing/{uuid}/paymentinfo/{paymentInfoId}/transactions
Description: Do a payment process with saved payment information.

GET **GetTransactions**

URL: /billing/{uuid}/transactions
Description: Gets a user’s transaction list.

GET **GetTransactionCount**

URL: /billing/transactions/{transactionDate}/count
Description: Gets a transaction count.

GET **GetTransactionHistory**

URL: /billing/transactions/{transactionDate}/history
Description: Gets a transaction history list.

POST **Payout Transaction**

URL: /billing/{uuid}/payout
Description: Does the Payout Process.

DELETE **RefundTransaction**

URL: /billing/{uuid}/transactions?referenceTransactionId={referenceTransactionId}&note={note}
Description: Does the Refund transaction with Reference Transaction.

PUT **RejectTransaction**

URL: /billing/{uuid}/transactions/{referenceTransactionId}/status
Description: Does Reject Transaction process.

Example
Tech Documentation

Secure Payment APIs v5.51

Released: **2011-01-10** | Sprint: 5.51
Security classification: **PUBLIC**

Endpoint Index

Example Secure Payment APIs Security: Encrypted Data

The Example Secure Payment APIs protect all HTTPS Request and Response data according to PCI and PII standards with the AES-GCM encryption *(https://tools.ietf.org/html/rfc5288)* (Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) as a Transport Layer Security (TLS) authenticated encryption operation). Please see the Protocol Rules "AES-GCM encrypted request and response data" section for these and other Example Secure Payment API security-related details.

The following endpoints are used in the Payment APIs.

Account APIs

- **DELETE** **/account/{uuid}/orders/{orderId}** | Cancel User's Payment Information
- **POST** **/account/{uuid}/paymentinfo** | Create User's Payment Information
- **PUT** **/account/{uuid}/paymentinfo/{paymentInfoId}** | Update User's Payment Information
- **DELETE** **/account/{uuid}/paymentinfo/{paymentInfoId}** | Delete User's Payment Information
- **GET** **/account/{uuid}/paymentinfo** | Get User's Payment Information List
- **GET** **/account/{uuid}/paymentinfo/{paymentinfoId}** | Get User's Payment Information by paymentInfoID
- **POST** **/account/{uuid}/paymentinfo/{paymentinfoId}/verification** | Reverify Payment Information

Billing APIs

- **DELETE** **/billing/{uuid}/orders/{orderId}** | Cancel Transaction By Mobile Client OrderId
- **POST** **/billing/{uuid}/transactions/check** | Check Billing Transaction
- **POST** **/billing/{uuid}/transactions** | Create Billing Transaction with New Payment Information
- **POST** **/billing/{uuid}/paymentinfo/{paymentinfoid}/transactions** | Create Billing Transaction with paymentInfoID
- **GET** **/billing/{uuid}/transactions** | Get Billing Transaction
- **GET** **/billing/transactions/{transactionDate}/count** | Get Transaction Count
- **GET** **/billing/transactions/{transactionDate}/history** | Get Transaction History List
- **POST** **/billing/{uuid}/payout** | Payout Transaction

DELETE **/billing/{uuid}/transactions** | Refund Transaction

PUT **/billing/{uuid}/transactions/{referenceTransactionId}/status** | Reject Transaction

Example
Tech Documentation

Secure Payment APIs v5.51

Released: **2011-01-10** | Sprint: 5.51
Security classification: **PUBLIC**

Account APIs

Example Secure Payment APIs Security: Encrypted Data

The Example Secure Payment APIs protect all HTTPS Request and Response data according to PCI and PII standards with the AES-GCM encryption *(https://tools.ietf.org/html/rfc5288)* (Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) as a Transport Layer Security (TLS) authenticated encryption operation). Please see the Protocol Rules "AES-GCM encrypted request and response data" section for these and other Example Secure Payment API security-related details.

The APIs are listed with their Common Names followed by their `(CodeNames)`.

DELETE Cancel User's Payment Information `(CancelPaymentInfoByOrderId)`

Description
A server can invoke this action to delete a user's payment information.

Endpoint url
DELETE /account/{uuid}/orders/{orderId}

HTTPS Request data: Parameters
Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	36	M	User's unique identification
orderId	String	36	M	A unique order ID for payment info used in a reference request.

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
orderId	String	36	Unique OrderId for payment info used in a reference request.

HTTPS Request example: CancelPaymentInfoByOrderId [DELETE method]

```
https
account/2303135303932343135303
6323133626338363033333213836e14946e58b0485ae4ab000557942caa75e8
4708a6123/orders/230313530393234313530363231336263383630333332313
3836e14946e58b0485ae4ab000557942caa75e84708a6123
```

Encrypting Data and HTTPS Request [DELETE method]

```
https
DELETE
http(s)://{host}/account/230313530393234313530363231336263383630
333332313836e14946e58b0485ae4ab000557942caa75e84708a6123/orders/
230313530393234313530363231336263383630333332313836e14946e58b048
5ae4ab000557942caa75e84708a6123
```

```
"Authorization: PLTOKEN {TOKEN}"
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

```https
32303135303932343135303433323566623536346361323134388818b824cf62
c2d63feba0314c8b6cd3b328c2f02380787406779be1607fa68363bf1fbac60
a1b691de2f283ee2370f04f479287589b712576357c3256a2ba87f9c24d5948
bb91dd13211ed9e452247603a75b5f5672f6718242e78baf48524182e5f35ff
a58874f3692f9ca9011d84d70e8f74dad430bf8a58c8330fb0927e6048db45e
3b18544a2d780e005903a668bbfba9fa7f0b5522ba0719c4e643c9fea11563f
bd62d594c459d825f96ef53ed81aab2ec6a0657288c799e418e7e2d0451ca52
35a0d7dde3fe13f31cd6e7bd46470a58c2cb7372587b8e30684fa98c2b6ddf3
e06392bcbe41b80d107790ea58cc5b521156264ac9051b1b4aed931f9ba5b2b
be7e12b62de036f7cd1fb328bfe80c8781a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK

```json
{
  “data”: {
    “orderId”: “272d11bc-72d2-4ec2-99f8-1374daec3535”
  }
}
```

Failed Response

HTTPS STATUS CODE : 400 Not Found

```json
{
  "error": {
    "code": 903,
    "message": "This account payment information does not exist",
    "detail": null
  }
}
```

POST Create User's Payment Information (CreatePaymentInfo)

Description

A server can invoke this action to create a user's payment information.

Endpoint url

POST /account/{uuid}/paymentinfo

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	36	M	User's unique identification
udid	String	50	M	User's device identification.
orderId	String	36	M	A unique order ID.
billingInfo	Object	N/A	M	Billing Information Object
firstName	String	50	M	User's First Name
lastName	String	50	M	User's Last Name
middleInitial	String	50	O	User's Middle Name
dateOfBirth	String	8	M	User's date of birth. Format = "YYYYMMDD" Example: "19870704"
phoneNumber	String	20	M	User's phone number.
emailAddress	String	200	M	User's Email Address
addressInfo	Object	N/A	M	Address Information Object
street	String	256	M	Street address
street2	String	50	O	Detailed Street address
city	String	50	M	City
state	String	50	O	For United States and Canada, 2-Characters state code should be sent. Other countries have no restrictions. ISO 3166-2:US *(https://en.wikipedia.org/wiki/ISO_3166-2:US)* ISO 3166-2:CA *(https://en.wikipedia.org/wiki/ISO_3166-2:CA)*
postalCode	String	20	M	Postal code
countryCode	String	2	M	Country code ("GB"): ISO 3166-2 *(https://en.wikipedia.org/wiki/ISO_3166-2)*
paymentInfo	Object	N/A	M	Payment Information Object
paymentType	String	50	M	Payment Type: "CreditCard"
isDefault	Boolean	1	O	Indicates if a user has made this the default payment method: true or false.
card	Object	N/A	M	Card Information Object
cardNum	String	16	M	Card Number
cardExpYear	String	4	M	Card Expiry Year
cardExpMonth	String	2	M	Card Expiry Month
cardCVVToken	String	20	M	SAKA Token for decryption Card Verification Value
cardHolder	String	50	M	Card Holder
cardNickName	String	20	O	Card Nick Name

Parameter	Type	Size	M/O	Description
accessCountryCode	String	2	M	User's access country code based on GEO IP: ISO 3166-2 *(https://en.wikipedia.org/wiki/ISO_3166-2)*
ipAddress	String	50	M	User's IP Address

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
uuid	String	36	User's Unique ID
paymentInfoID	String	36	Payment Information ID
registeredDate	String	20	Registration Date based on UTC "MM/dd/yyyy HH:mm:ss" format. Example: "12/11/2015 10:01:01"

HTTPS Request example: CreatePaymentInfo [POST method]

```json
{
  "orderId": "251153cb-de49-4afb-9fb5-acb33e332b26",
  "udid": "2083718938012321038243",
  "billingInfo": {
    "firstName": "John",
    "lastName": "Doe",
    "dateOfBirth":"19870704",
    "phoneNumber":"1234567890",
    "emailAddress": "abc@xyz.com",
    "addressInfo": {
      "street": "18th Floore Prtlan House",
        "street2": "Bressenden Place",
      "city": "Victoria",
      "state": "London",
      "postalCode": "SW1E 5RS",
      "countryCode": "GB"
    }
  },
  "paymentInfo": {
    "paymentType": "CreditCard",
    "isDefault": true,
    "card": {
      "cardNum": "4111111111111111",
      "cardExpYear": "2018",
      "cardExpMonth": "08",
      "cardHolder": "John Doe",
      "cardNickName": "Card1",
      "cardCvvToken": "1000000000000016"
    }
  },
  "accessCountryCode": "GB",
  "ipAddress": "127.0.0.1"
}
```

Encrypting Data and HTTPS Request [POST method]

```https
POST
http(s)://{host}/account/32303136303432383136343234356330633438 3
53933386466644aa448bc71a97b5267514c0b4c0ad7d54463f3a44f913391af5
320bb01f188a11b501cf5ffd075ffb1278f1c3a3d8ccc9b934b1b/paymentinfo

"Content-Type: application/x-www-form-urlencoded"
```

```
"Authorization: PLTOKEN {TOKEN}"

data=
3230313530393234313530363231336263383630333332313836e14946e58b04
85ae4ab000557942caa75e84708a6123ac21a9945845a07d557a6bee23a9b01d
b139e3d5895bae320476cc665ac7b5864baeab3a806570b5e4f9b2ba8a408576
43913f93e6534753ab08be5815a0d170d0f679a3c3d838ae885e80f01f40bd45
a0d506858e6c97ca0e55d0aa565c79b3a5e34ab81c4e7c9b754557231be0760e
8041eb7cbba8a0e9db51a93c99917e39cf1cda93412b1c20795ce171e9b9e316
006cb7830085b0434eb2c90a12d749f975c09f3fe7ccc11242c7452edb5325ba
a9a852298ab059d40e09a3b653e64c953adddf04f6761747d66827820035c584
52c2fa9c72af00a7caa79a1d39ca467acc7dd05b58ed2f36ca02f20fd4523fcc
84361ff7f7cd81985ff283f71cbd4cae958e6055a127cba408b23d3bee884e04
a986eb94a943442f428e4daa207af947d3a03faa77f47b2ca7f06382af36567e
3cb6ead4f19ffdfc850111971e78b68453abea1aa650a2eacf00505dbc84c1b6
aa2b2c1f9814bb34997e8e13906810ea92533c823df4a2ebfd4fcf7683328b45
2a360736fd65d5d1ddad22466078bd74aa9705e7b58e58a846ccb46b9fa9ea81
06a6b520f596bc44407541d8bfbc54f95797895e758076864502aef033b7223d
b03ad35d0d1934d67e18dd58a0fba2da2a323bae7fc971829eb68edfc69e0ae8
73cd1df8099509
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

https

```
3230313530393234313530343332356662353634636132313438818b824cf62
c2d63feba0314c8b6cd3b328c2f02380787406779be1607fa68363bf1fbac60
a1b691de2f283ee2370f04f479287589b712576357c3256a2ba87f9c24d5948
bb91dd13211ed9e452247603a75b5f5672f6718242e78baf48524182e5f35ff
a58874f3692f9ca9011d84d70e8f74dad430bf8a58c8330fb0927e6048db45e
3b18544a2d780e005903a668bbfba9fa7f0b5522ba0719c4e643c9fea11563f
bd62d594c459d825f96ef53ed81aab2ec6a0657288c799e418e7e2d0451ca52
35a0d7dde3fe13f31cd6e7bd46470a58c2cb7372587b8e30684fa98c2b6ddf3
e06392bcbe41b80d107790ea58cc5b521156264ac9051b1b4aed931f9ba5b2b
be7e12b62de036f7cd1fb328bfe80c8781a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK

json

```json
{
  "data": {
    "uuid": "2a50a2e4-a93a-4823-9674-c63aba792325",
    "paymentInfoId": "91534985-ff82-42dc-aebe-72484455f47e",
    "registeredDate": "04/25/2016 04:25:28"
  }
}
```

Failed Response

HTTPS STATUS CODE : 500 Internal Server Error

json

```json
{
  "error": {
    "code": 933,
    "message": "Failed to create payment information.",
    "detail": "[23] Invalid credit card information. Please re-enter. (Invalid account number)"
  }
}
```

PUT Update User's Payment Information (UpdatePaymentInfo)

Description

A server can invoke this action to update a user's payment information.

Endpoint url

/account/{uuid}/paymentinfo/{paymentInfoId}

HTTPS Method

PUT

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters udid

Parameter	Type	Size	M/O	Description
uuid	String	36	M	Encrypted user's unique ID.
String	50	M	User's device identification.	
paymentInfoId	String	N/A	M	Encrypted user's unique payment information ID.
orderId	String	36	M	A unique order ID.
billingInfo	Object	N/A	M	Billing Information Object
firstName	String	50	M	User's First Name
lastName	String	50	M	User's Last Name
middleInitial	String	50	O	User's Middle Name
dateOfBirth	String	8	M	User's date of birth. Format = "YYYYMMDD" Example: "19870704"
phoneNumber	String	20	M	User's phone number.
emailAddress	String	200	M	User's Email Address.
addressInfo	Object	N/A	M	Address Information Object.
street	String	256	M	Street address.
street2	String	50	O	Detailed Street address
city	String	50	M	City
state	String	50	O	For United States and Canada, 2-Characters state code should be sent. Other countries have no restrictions. ISO 3166-2:US *(https://en.wikipedia.org/wiki/ISO_3166-2:US)* ISO 3166-2:CA *(https://en.wikipedia.org/wiki/ISO_3166-2:CA)*
postalCode	String	20	M	Postal code
countryCode	String	2	M	Country code ("GB"): ISO 3166-2 *(https://en.wikipedia.org/wiki/ISO_3166-2)*
paymentInfo	Object	N/A	M	Payment Information Object
paymentType	String	50	M	Payment Type: "CreditCard"
isDefault	Boolean	1	O	Indicates if a user has made this the default payment method: true or false.
card	Object	N/A	M	Card Information Object.
cardExpYear	String	4	M	Card Expiry Year.
cardExpMonth	String	2	M	Card Expiry Month.

Parameter	Type	Size	M/O	Description
cardCVVToken	String	20	M	SAKA Token for decryption of the Card Verification Value.
cardHolder	String	50	M	Card Holder
cardNickName	String	20	O	Card Nick Name
accessCountryCode	String	2	M	User's access country code based on GEO IP. See ISO 3166-2 *(https://en.wikipedia.org/wiki/ISO_3166-2)*
ipAddress	String	50	M	User's IP Address

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
uuid	String	200	User's Unique ID
paymentInfoID	String	36	Payment Information ID
status	String	20	Status of payment information: "ACTIVE", "INACTIVE", "FRAUD"
updatedDate	String	20	Updated Date based on UTC "MM/dd/yyyy HH:mm:ss" format. For example: "12/11/2015 10:01:01"

HTTPS Request example: UpdatePaymentInfo [PUT method]

```json
{
  "orderId": "4e260748-60d6-4985-80da-bfdf481bffdf",
  "udid": "2083718938012321038243",
  "billingInfo": {
    "firstName": "John",
    "lastName": "Doe",
    "dateOfBirth":"19870704",
    "phoneNumber":"1234567890",
    "emailAddress": "abc@xyz.com",
    "addressInfo": {
      "street": "18th Floore Prtlan House",
      "street2": "Bressenden Place",
      "city": "Victoria",
      "state": "London",
      "postalCode": "SW1E 5RS",
      "countryCode": "GB"
    }
  },
  "paymentInfo": {
    "paymentType": "CreditCard",
    "isDefault": false,
    "card": {
      "cardExpYear": "2018",
      "cardExpMonth": "08",
      "cardHolder": "John Doe",
      "cardNickName": "Card1",
      "cardCvvToken": "1000000000000016"
    }
  },
  "accessCountryCode": "GB",
  "ipAddress": "127.0.0.1"
}
```

Encrypting Data and HTTPS Request [PUT method]

```https
PUT
http(s)://{host}/account/230313530393234
313530363231336263383630333332313836e149
46e58b0485ae4ab000557942caa75e84708a6123
/paymentinfo/2303135303932343135303632313
3362633836303333323133836e14946e58b0485ae
4ab000557942caa75e84708a6123

"Content-Type: application/x-www-form-urlencoded"
"Authorization: PLTOKEN {TOKEN}"

data=
323031353039323431353036323133362633383630
333332313836e14946e58b0485ae4ab000557942
caa75e84708a6123ac21a9945845a07d557a6bee
23a9b01db139e3d5895bae320476cc665ac7b586
4baeab3a806570b5e4f9b2ba8a40857643913f93
e6534753ab08be5815a0d170d0f679a3c3d838ae
885e80f01f40bd45a0d506858e6c97ca0e55d0aa
565c79b3a5e34ab81c4e7c9b754557231be0760e
8041eb7cbba8a0e9db51a93c99917e39cf1cda93
412b1c20795ce171e9b9e316006cb7830085b043
4eb2c90a12d749f975c09f3fe7ccc11242c7452e
db5325baa9a852298ab059d40e09a3b653e64c95
3adddf04f6761747d66827820035c58452c2fa9c
72af00a7caa79a1d39ca467acc7dd05b58ed2f36
ca02f20fd4523fcc84361ff7f7cd81985ff283f7
1cbd4cae958e6055a127cba408b23d3bee884e04
a986eb94a943442f428e4daa207af947d3a03faa
77f47b2ca7f06382af36567e3cb6ead4f19ffdfc
850111971e78b68453abea1aa650a2eacf00505d
bc84c1b6aa2b2c1f9814bb34997e8e13906810ea
92533c823df4a2ebfd4fcf7683328b452a360736
fd65d5d1ddad22466078bd74aa9705e7b58e58a8
46ccb46b9fa9ea8106a6b520f596bc44407541d8
bfbc54f95797895e758076864502aef033b7223d
b03ad35d0d1934d67e18dd58a0fba2da2a323bae
7fc971829eb68edfc69e0ae873cd1df8099509
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

```https
3230313530393234313530343332356662353634
636132313438818b824cf62c2d63feba0314c8b6
cd3b328c2f02380787406779be1607fa68363bf1
fbac60a1b691de2f283ee2370f04f479287589b7
12576357c3256a2ba87f9c24d5948bb91dd13211
ed9e452247603a75b5f5672f6718242e78baf485
24182e5f35ffa58874f3692f9ca9011d84d70e8f
74dad430bf8a58c8330fb0927e6048db45e3b185
44a2d780e005903a668bbfba9fa7f0b5522ba071
9c4e643c9fea11563fbd62d594c459d825f96ef5
3ed81aab2ec6a0657288c799e418e7e2d0451ca5
235a0d7dde3fe13f31cd6e7bd46470a58c2cb737
2587b8e30684fa98c2b6ddf3e06392bcbe41b80d
107790ea58cc5b521156264ac9051b1b4aed931f
9ba5b2bbe7e12b62de036f7cd1fb328bfe80c878
1a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK

```json
{
  "data": {
    "uuid": "89396B99-5A57-4ECA-96C9-60ED109F8C8A",
    "paymentInfoId": "49660FF3-8D52-4641-9770-FDD199E1CD18",
    "status": "ACTIVE",
    "updatedDate": "12/11/2015 10:01:01"
  }
}
```

Failed Response

HTTPS STATUS CODE : 500 Internal Server Error

```json
{
  "error": {
    "code": 975,
    "message": "An unexpected error occurred during update payment information. Please try again.",
    "detail": "Cannot authorize this card information."
  }
}
```

DELETE Delete User's Payment Information (DeletePaymentInfo)

Description

A server can invoke this action to delete a user's payment information.

Endpoint url

/account/{uuid}/paymentinfo/{paymentInfoId}

HTTPS Method

DELETE

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	N/A	M	Encrypted user's unique ID
paymentInfoId	String	N/A	M	Encrypted user's unique payment information ID

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
uuid	String	36	User's unique ID.

HTTPS Request example: DeletePaymentInfo [DELETE method]

```
https
account/
230313530393234313530363231336263383630333
332313836e14946e58b0485ae4ab000557942caa75
e84708a6123/paymentinfo/230313530393234313
530363231336263383630333332313836e14946e58
b0485ae4ab000557942caa75e84708a6123
```

Encrypting Data and HTTPS Request [DELETE method]

```
https
DELETE
https://{host}/account/230313530393234313530363 2
313362633836303333323138 36e14946e58b0485ae4ab000
557942caa75e84708a6123/paymentinfo/2303135303932
343135303632313362633836303333323138 36e14946e58b
0485ae4ab000557942caa75e84708a6123

"Authorization: PLTOKEN {TOKEN}"
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

```
https
323031353039323431353034333235666235363 4
636132313438818b824cf62c2d63feba0314c8b6
cd3b328c2f02380787406779be1607fa68363bf1
fbac60a1b691de2f283ee2370f04f479287589b7
12576357c3256a2ba87f9c24d5948bb91dd13211
ed9e452247603a75b5f5672f6718242e78baf485
24182e5f35ffa58874f3692f9ca9011d84d70e8f
74dad430bf8a58c8330fb0927e6048db45e3b185
44a2d780e005903a668bbfba9fa7f0b5522ba071
9c4e643c9fea11563fbd62d594c459d825f96ef5
```

```
3ed81aab2ec6a0657288c799e418e7e2d0451ca5
235a0d7dde3fe13f31cd6e7bd46470a58c2cb737
2587b8e30684fa98c2b6ddf3e06392bcbe41b80d
107790ea58cc5b521156264ac9051b1b4aed931f
9ba5b2bbe7e12b62de036f7cd1fb328bfe80c878
1a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK

json

```json
{
  "data": {
    "uuid": "272d11bc-72d2-4ec2-99f8-1374daec3535"
  }
}
```

Failed Response

HTTPS STATUS CODE : 404 Not Found

json

```json
{
  "error": {
    "code": 903,
    "message": "This account payment information does not exist",
    "detail": null
  }
}
```

GET Get User's Payment Information List (GetPaymentInfo)

Description

A server can invoke this action to get the list of a user's payment information.

Endpoint url

/account/{uuid}/paymentinfo

HTTPS Method

GET

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	200	M	Encrypted user's unique ID

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
uuid	String	36	User's Unique ID
totalCount	Integer	4	Total Count
paymentInfos	Array (Object)	N/A	Array of Payment Information object
paymentInfoId	String	36	Payment Information ID
status	String	20	Status of Payment Information: "ACTIVE", "INACTIVE", "FRAUD"
paymentInfo	Array (Object)	N/A	Payment Information Object
paymentType	String	50	Payment type: "CreditCard"

HTTPS Request example: GetPaymentInfo [GET method]

```
https
/account
/32303135303932343135303632313362633836303 3
3332313836e14946e58b0485ae4ab000557942caa75
e84708a6123a/paymentinfo
```

Encrypting Data and HTTPS Request [GET method]

```
https
GET
https://{host}/account/230313530393234313530
3632313362633836303333332313836e14946e58b048
5ae4ab000557942caa75e84708a6123/paymentinfo

"Authorization: PLTOKEN {TOKEN}"
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

```
https
32303135303932343135303433323566623536 34
636132313438818b824cf62c2d63feba0314c8b6
cd3b328c2f02380787406779be1607fa68363bf1
fbac60a1b691de2f283ee2370f04f479287589b7
12576357c3256a2ba87f9c24d5948bb91dd13211
ed9e452247603a75b5f5672f6718242e78baf485
```

```
24182e5f35ffa58874f3692f9ca9011d84d70e8f
74dad430bf8a58c8330fb0927e6048db45e3b185
44a2d780e005903a668bbfba9fa7f0b5522ba071
9c4e643c9fea11563fbd62d594c459d825f96ef5
3ed81aab2ec6a0657288c799e418e7e2d0451ca5
235a0d7dde3fe13f31cd6e7bd46470a58c2cb737
2587b8e30684fa98c2b6ddf3e06392bcbe41b80d
107790ea58cc5b521156264ac9051b1b4aed931f
9ba5b2bbe7e12b62de036f7cd1fb328bfe80c878
1a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK

json


```json
{
  "data": {
    "uuid": "dc79042f-321e-42df-9e12-25fc2a4a8f1a",
    "totalCount": 3,
    "paymentInfos": [
    {
      "paymentInfoId": "bca80e4a-737d-48e2-afb0-86031a0af886",
      "status": "ACTIVE",
      "paymentInfo": {
        "paymentType": "CreditCard"
      }
    },
    {
    "paymentInfoId": "8bae8471-0785-4c39-b016-086f8d54d58a",
    "status": "ACTIVE",
    "paymentInfo": {
      "paymentType": "CreditCard"
    }
    },
    {
    "paymentInfoId": "7eb5db22-c716-46f6-95d1-f54dc63fa3e1",
    "status": "ACTIVE",
    "paymentInfo": {
      "paymentType": "CreditCard"
    }
   }
  ]
 }
}
```


Failed Response

HTTPS STATUS CODE : 404 Not Found

json


```json
{
  "error": {
    "code": 903,
    "message": "This account's payment information does not exist",
    "detail": null
  }
}
```

GET Get User's Payment Information by paymentInfoID (GetPaymentinfoByPaymentInfoID)

Description

A server can invoke this action to get specific payment information by uuid and paymentInfoID.

Endpoint url

/account/{uuid}/paymentinfo/{paymentinfoId}

HTTPS Method

GET

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	N/A	M	Encrypted user's unique ID.
paymentInfoId	String	N/A	M	Encrypted user's unique payment information ID.

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
uuid	String	36	User's unique ID.
paymentInfoID	String	36	Payment Information ID
status	String	20	Status of Payment Information: "ACTIVE", "INACTIVE", "FRAUD"
billingInfo	Object	N/A	Billing Information Object
firstName	String	50	User's First Name
lastName	String	50	User's Last Name
middleInitial	String	50	User's Middle Name
dateOfBirth	String	8	User's date of birth. Format = "YYYYMMDD" Example: "19870704"
phoneNumber	String	20	User's phone number.
emailAddress	String	200	User's Email Address
addressInfo	Object	N/A	Address Information Object
street	String	256	Street address
street2	String	50	Detailed Street address
city	String	50	City
state	String	50	For United States and Canada, 2-Characters state code should be sent. Other countries have no restrictions. ISO 3166-2:US *(https://en.wikipedia.org/wiki/ISO_3166-2:US)* ISO 3166-2:CA *(https://en.wikipedia.org/wiki/ISO_3166-2:CA)*
postalCode	String	20	Postal code
countryCode	String	2	Country code ("GB"): ISO 3166-2 *(https://en.wikipedia.org/wiki/ISO_3166-2)*
paymentInfo	Array (Object)	N/A	Payment Information object.
paymentType	String	50	Payment Type: "CreditCard".
card	Object	N/A	Card Information object.
cardNum	String	16	The card number.
cardBrand	String	50	The card brand.

JSON Property	Type	Size	Description
cardLastNum	String	4	Last 4 digits of the card number.
cardExpYear	String	4	Card Expiry Year.
cardExpMonth	String	2	Card Expiry Month.
cardHolder	String	50	Card Holder.

HTTPS Request example: GetPaymentInfo [GET method]

```https
/account/b7f7ea57-6503-4b5b-9f64-c437f0f78563
/paymentinfo/a55a3851-8a62-48f6-b9bb-91a853ac97b5
```

Encrypting Data and HTTPS Request [GET method]

```https
GET
https://{host}/account/32303135303932343135303934326234
6336434363931636134991a219bba46116e3deecfe485f650a558f86
1cafb503295930eccc7ce11b99f57524ef67713f45582a5eb07e9fe8
f1d9e19c933/paymentinfo/32303135303932343135303934326234
6633643436393163613499 1a219bba46116e3deecfe485f650a558f8
61cafb50295930eccc7ce11b99f57524ef67713f45582a

"Authorization: PLTOKEN {TOKEN}"
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

```https
32303135303932343135303433323566623536 34
636132313438818b824cf62c2d63feba0314c8b6
cd3b328c2f02380787406779be1607fa68363bf1
fbac60a1b691de2f283ee2370f04f479287589b7
12576357c3256a2ba87f9c24d5948bb91dd13211
ed9e452247603a75b5f5672f6718242e78baf485
24182e5f35ffa58874f3692f9ca9011d84d70e8f
74dad430bf8a58c8330fb0927e6048db45e3b185
44a2d780e005903a668bbfba9fa7f0b5522ba071
9c4e643c9fea11563fbd62d594c459d825f96ef5
3ed81aab2ec6a0657288c799e418e7e2d0451ca5
235a0d7dde3fe13f31cd6e7bd46470a58c2cb737
2587b8e30684fa98c2b6ddf3e06392bcbe41b80d
107790ea58cc5b521156264ac9051b1b4aed931f
9ba5b2bbe7e12b62de036f7cd1fb328bfe80c878
1a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK

```json
{
  "data": {
    "uuid": "2a50a2e4-a93a-4823-9674-c63aba792325",
    "paymentInfoId": "28da53f2-e8d2-49e2-b63c-31e2d21dd343",
    "status": "ACTIVE",
    "billingInfo": {
      "firstName": "John",
```

```
      "lastName": "Doe",
      "dateOfBirth":"19870704",
      "phoneNumber":"1234567890",
      "emailAddress": "abc@xyz.com",
      "addressInfo": {
        "street": "18th Floore Prtlan House",
        "street2": "Bressenden Place",
        "city": "Victoria",
        "state": "London",
        "postalCode": "SW1E 5RS",
        "countryCode": "GB"
      }
    },
    "paymentInfo": {
      "paymentType": "CreditCard",
      "card": {
        "cardNum": "4111111111111111",
        "cardExpYear": "2018",
        "cardExpMonth": "08",
        "cardHolder": "John Doe",
        "cardNickName": "Card1",
        "cardBrand": "VPAY"
      }
    }
  }
}
```

Failed Response

HTTPS STATUS CODE : 404 Not Found

json

```
{
  "error": {
    "code": 903,
    "message": "This account's payment information does not exist.",
    "detail": null
  }
}
```

GET Get User's Card Expiries (GetCardExpiries)

Description

A server can invoke this action to get a list of a user's expired credit cards.

Endpoint url

/account/{uuid}/paymentinfo/expiries

HTTPS Method

GET

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	N/A	M	Encrypted user's unique ID.

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
uuid	String	36	User's Unique ID.
totalCount	Integer	4	Total Count.
paymentInfos	Array	N/A	Array of the Payment Information Object.
paymentInfoId	String	36	Payment Information ID.
paymentInfo	Object	N/A	Payment Information Object.
paymentType	String	50	Payment type ("CreditCard")
card	Object	N/A	Payment Information Object
cardExpYear	String	4	Expired Date: Year (YYYY) format. Example: "2020"
cardExpMonth	String	2	Expired Date: Month (MM) format. Example: "04"

HTTPS Request example: GetCardExpiries [GET method]

```
https
/account/32303135303932343135303934326234663364343639316361349
23133626338363033333231383 6e14946e58b0
485ae4ab000557942caa75e84708a6123a/
paymentinfo/expiries
```

Encrypting Data and HTTPS Request [GET method]

```
https
GET
http(s)://{host}/account/32303135303932
343135303934326234663364343639316361349
91a219bba46116e3deecfe485f650a558f861ca
fb503295930eccc7ce11b99f57524ef67713f45
582a5eb07e9fe8f1d9e19c933/paymentinfo/
expiries
"Authorization: PLTOKEN {TOKEN}"
```

Successful *Encrypted* Response

```
HTTPS STATUS CODE : 200 OK
```

```https
3230313530393234313530343332356662353634636132313438818b824cf62
c2d63feba0314c8b6cd3b328c2f02380787406779be1607fa68363bf1fbac60
a1b691de2f283ee2370f04f479287589b712576357c3256a2ba87f9c24d5948
bb91dd13211ed9e452247603a75b5f5672f6718242e78baf48524182e5f35ff
a58874f3692f9ca9011d84d70e8f74dad430bf8a58c8330fb0927e6048db45e
3b18544a2d780e005903a668bbfba9fa7f0b5522ba0719c4e643c9fea11563f
bd62d594c459d825f96ef53ed81aab2ec6a0657288c799e418e7e2d0451ca52
35a0d7dde3fe13f31cd6e7bd46470a58c2cb7372587b8e30684fa98c2b6ddf3
e06392bcbe41b80d107790ea58cc5b521156264ac9051b1b4aed931f9ba5b2b
be7e12b62de036f7cd1fb328bfe80c8781a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK

```json
{
  "data": {
    "uuid": "dc79042f-321e-42df-9e12-25fc2a4a8f1a",
    "totalCount": 3,
    "paymentInfos": [
      {
        "paymentInfoId": "bca80e4a-737d-48e2-afb0-86031a0af886",
        "paymentInfo": {
          "paymentType": "CreditCard",
          "card": {
            "cardExpYear": "2019",
            "cardExpMonth": "08"
          }
        }
      },
      {
        "paymentInfoId": "8bae8471-0785-4c39-b016-086f8d54d58a",
        "paymentInfo": {
          "paymentType": "CreditCard",
          "card": {
            "cardExpYear": "2018",
            "cardExpMonth": "08"
          }
        }
      },
      {
        "paymentInfoId": "7eb5db22-c716-46f6-95d1-f54dc63fa3e1",
        "paymentInfo": {
          "paymentType": "CreditCard",
          "card": {
            "cardExpYear": "2018",
            "cardExpMonth": "08"
          }
        }
      }
    ]
  }
}
```

Failed Response

HTTPS STATUS CODE : 404Not Found

```json
{
  "error": {
    "code": 903,
    "message": "This account payment information does not exist.",
    "detail": null
  }
}
```

POST Reverify User's Payment Information (ReVerifyPaymentInfo)

Description

A server can invoke this action to re-verify payment information by uuid and paymentInfoID.

Endpoint url

/account/{uuid}/paymentinfo/{paymentinfoId}/verification

HTTPS Method

POST

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	N/A	M	Encrypted user's unique ID
paymentInfoId	String	N/A	M	Encrypted user's unique payment information ID
orderId	String	36	M	A unique order ID.
udid	String	50	M	User's device identification.
paymentInfo	Object	N/A	M	Payment Information Object
paymentType	String	50	M	Payment Type: "Credit Card"
card	Object	N/A	M	Credit Information Object
cardCVVToken	String	20	M	SAKA Token for decryption Card Verification Value
accessCountryCode	String	2	M	User's access country code based on GEO IP. See ISO 3166-2 *(https://en.wikipedia.org/wiki/ISO_3166-2)*
ipAddress	String	50	M	IP Address

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
uuid	String	200	User's Unique ID
paymentInfoId	String	36	Payment Information ID
status	String	20	Status of Payment Information: "ACTIVE", "INACTIVE", "FRAUD"

HTTPS Request example: ReVerifyPaymentInfo [POST method]

```json
{
  "orderId": "f98a9325-89fb-4aa0-8ac9-30869b5f1a79",
  "udid": "2083718938012321038243",
  "paymentInfo": {
    "paymentType": "CreditCard",
    "card": {
      "cardCvvToken": "1000000000000183"
    }
  },
  "accessCountryCode": "KR",
  "ipAddress": "127.0.0.1"
}
```

Encrypting Data and HTTPS Request [POST method]

```
https
POST
http(s)://{host}/account/323031363034323
8313634323435633063343835393338646664 4aa
448bc71a97b5267514c0b4c0ad7d54463f3a44f9
13391af5320bb01f188a11b501cf5ffd075ffb12
78f1c3a3d8ccc9b934b1b/paymentinfo/323031
363034323831363432343530343832613164663 4
363561d29d6dc31636226e97c39d92fb80621a63
8e39756a223b2f750216b05cf6eb5c9dde6b7611
b48cf47725b16e25a06041aed292c1/verification

"Content-Type: application/x-www-form-urlencoded"
"Authorization: PLTOKEN {TOKEN}"

data=
323031363034323831363432343532363931383 9
6562623231325be9680a1f756a284c8899774b1f
b34ea880558f34f7f01b65ebb14cd1a48a958c23
a823ba826ebff80ef83fd832879114fa6101e8cb
2001f51ed56574836927de54aced12591c95ff9c
8eb11aca4478f0d4bd7534549e2c580136014e3c
92e3fe12deaa7a7344449e065dcd52eb413b369c
12db2e419c6de0d08b43ace48b95e85fa73f2fbc
a061e33842d6593697c85460b7ba0cfaf04dee3e
fcc77468461ccd4d03435236f5753ee3e4d5725a
e73bdb96f0e105
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

```
https
323031353039323431353034333235666235363 4
636132313438818b824cf62c2d63feba0314c8b6
cd3b328c2f02380787406779be1607fa68363bf1
fbac60a1b691de2f283ee2370f04f479287589b7
12576357c3256a2ba87f9c24d5948bb91dd13211
ed9e452247603a75b5f5672f6718242e78baf485
24182e5f35ffa58874f3692f9ca9011d84d70e8f
74dad430bf8a58c8330fb0927e6048db45e3b185
44a2d780e005903a668bbfba9fa7f0b5522ba071
9c4e643c9fea11563fbd62d594c459d825f96ef5
3ed81aab2ec6a0657288c799e418e7e2d0451ca5
235a0d7dde3fe13f31cd6e7bd46470a58c2cb737
2587b8e30684fa98c2b6ddf3e06392bcbe41b80d
107790ea58cc5b521156264ac9051b1b4aed931f
9ba5b2bbe7e12b62de036f7cd1fb328bfe80c878
1a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK

```
json
{
  "data": {
    "uuid": "c3f7a09e-0ff8-4012-877f-8614fc570d42",
    "paymentInfoId": "f4fa63c9-473c-4d96-b173-b304c656b287",
    "status": "ACTIVE"

```

```
  }
}
```

Failed Response

HTTPS STATUS CODE : 404 Not Found

```json
{
  "error": {
    "code": 906,
    "message": "cvv information does not exist.",
    "detail": null
  }
}
```

Example
Tech Documentation

Secure Payment APIs v5.51

Released: **2011-01-10** | Sprint: 5.51
Security classification: **PUBLIC**

Billing APIs

Example Secure Payment APIs Security: Encrypted Data

The Example Secure Payment APIs protect all HTTPS Request and Response data according to PCI and PII standards with the AES-GCM encryption *(https://tools.ietf.org/html/rfc5288)* (Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) as a Transport Layer Security (TLS) authenticated encryption operation). Please see the Protocol Rules "AES-GCM encrypted request and response data" section for these and other Example Secure Payment API security-related details.

The APIs are listed with their Common Names followed by their `(CodeNames)`.

DELETE Cancel Transaction By Mobile Client OrderId `(CancelTransactionByOrderId)`

Description
A server can invoke this action to reject a transaction.

Endpoint url
/billing/{uuid}/orders/{orderId}

HTTPS Method
DELETE

HTTPS Request data: Parameters
Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	36	M	User's unique identification
orderId	String	36	M	Unique OrderId for a transaction by the client side. Note: Should be the same as the Check Billing Transaction's orderId.

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
orderId	String	36	The unique OrderId for a transaction which is used in a reference deposit.

HTTPS Request example: CancelTransaction [DELETE method]

```
https
billing/32303136313030363139353331353063636462326536313062331addcf82a6c8c8ed14c61aaa402555576c6
73188b933bcc163dfc028be4556519d629cfaee9b78c6e5c3140252a782b29ce3abe0/transactions?data=32303136
313030363139353331356261346336306535373333335394a56918a7af6f732864da9ea7d18a37d5ea4b440e95ef4e
541d9ce99edb4980396370adb09673c756ecf0ab3dedad3d156dfdfa9d072f8df9a219e93a011472a7632dc0ff3344
e92155aace8e5ff4c164f2feae28acc6fec00099214aadf2be3f68b8f487a8a9f3b4e106eb3f93f44c3a653db22166679
5a5dd07d13a830
```

Encrypting Data and HTTPS Request [POST method]

```
https

DELETE
http(s)://{host}/billing/32303136313030363139353331353063636462326536313062331addcf82a6c8c8ed14c61a
aa402555576c673188b933bcc163dfc028be4556519d629cfaee9b78c6e5c3140252a782b29ce3abe0/transactions?
data=32303136313030363139353331356261346336306535373333335394a56918a7af6f732864da9ea7d18a37d5e
a4b440e95ef4e541d9ce99edb4980396370adb09673c756ecf0ab3dedad3d156dfdfa9d072f8df9a219e93a011472a
7632dc0ff3344e92155aace8e5ff4c164f2feae28acc6fec00099214aadf2be3f68b8f487a8a9f3b4e106eb3f93f44c3a6
53db221666795a5dd07d13a830

"Authorization: PLTOKEN {TOKEN}"
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

```
https

323031353039323431353034333235666235363463613231 3
438818b824cf62c2d63feba0314c8b6cd3b328c2f02380787
406779be1607fa68363bf1fbac60a1b691de2f283ee2370f0
4f479287589b712576357c3256a2ba87f9c24d5948bb91dd1
3211ed9e452247603a75b5f5672f6718242e78baf48524182
e5f35ffa58874f3692f9ca9011d84d70e8f74dad430bf8a58
c8330fb0927e6048db45e3b18544a2d780e005903a668bbfb
a9fa7f0b5522ba0719c4e643c9fea11563fbd62d594c459d8
25f96ef53ed81aab2ec6a0657288c799e418e7e2d0451ca52
35a0d7dde3fe13f31cd6e7bd46470a58c2cb7372587b8e306
84fa98c2b6ddf3e06392bcbe41b80d107790ea58cc5b52115
6264ac9051b1b4aed931f9ba5b2bbe7e12b62de036f7cd1fb
328bfe80c8781a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK

```json
{
  "data": {
    "orderId": "139f0344-0641-4d9f-a47f-e695d05842a9"
  }
}
```

Failed Response [HTTPS + JSON]

HTTPS STATUS CODE : 500 Internal Server Error

```json
{
  "error": {
    "code": 947,
    "message": "Failed to reject payment.",
    "detail": Not existing payment request detail."
  }
}
```

POST **Check Billing Transaction** (`CheckTransaction`)

Description

A server can invoke this action to check transaction information before creating a new billing transaction.

- **Important note:** In the case of a card payment, if the card is enrolled 3DSecure it should proceed to the 3DSecure process.
- Please refer to the 3DSecure topic, section "Integration 3D Secure".

Endpoint url

/billing/{uuid}/transactions/check

HTTPS Method

POST

HTTPS Request data: Parameters

Note: **M/O/C** = Mandatory, Optional, or Conditional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	36	M	User's unique identification.
udid	String	50	M	User's unique device identification.
orderId	String	36	M	Unique OrderId for a transaction by the client side.
transaction	Object	N/A	M	Transaction Object.
applicationId	String	36	M	Unique ID for a casino (application).
applicationVersion	String	10	M	Casino (application) version.
amount	Double	8	M	The requested amount.
currencyCode	String	3	M	The currency code. See: ISO 4217 *(https://en.wikipedia.org/wiki/ISO_4217)*
extCasinoUserId	String	N/A	O	Corresponds to a casino's user id.
extCasinoTransId	String	N/A	O	Corresponds to a casino's transaction id.
extCasinoPaymentGatewayId	String	N/A	O	Corresponds to a casino's wallet id.
categoryNames	Array of strings	N/A	M	Category names of a casino, for example "Poker", "Bet"
casinoId	String	50	M	UUID of a casino.
accessCountryCode	String	2	M	User's access country code based on GEO IP. See ISO 3166-2 *(https://en.wikipedia.org/wiki/ISO_3166-2)*
ipAddress	String	50	M	User's IP address.
existingFlag	String	1	M	Existing Payment Info ID or New Payment information. "Y" = Existing "N" = New payment information.
gps	Object	N/A	O	Client's GPS information.
latitude	String	10	O	Latitude of the GPS. Example: "37.554738"
longitude	String	10	O	Longitude of the GPS. Example: "126.970680"
existingFlag is "Y"				
paymentInfoId	String	1	C	User's payment information.

Parameter	Type	Size	M/O	Description
paymentInfo	Object	N/A	M	Payment information object.
paymentType	String	50	M	Payment type ("CreditCard").
card	Object	N/A	M	Card Information Object.
cardCVVToken	String	20	M	SAKA token for the decryption Card Verification Value.
existingFlag is "N"				
saveFlag	String	1	C	Save user's payment information. "Y" = Save. "N" = Not save.
billingInfo	Object	N/A	M	Billing information object.
firstName	String	50	M	User's first name
lastName	String	50	M	User's last name.
middleInitial	String	50	O	User's middle name.
dateOfBirth	String	8	M	User's date of birth. Format is "YYYYMMDD". Example: 19870704
phoneNumber	String	20	M	User's phone number.
emailAddress	String	200	M	User's email address.
addressInfo	Object	N/A	M	Address information object.
street	String	256	M	Street address.
street2	String	50	O	Detailed street address.
city	String	50	M	City.
state	String	50	O	For United States and Canada, 2-Characters state code should be sent. Other countries have no restrictions. ISO 3166-2:US *(https://en.wikipedia.org/wiki/ISO_3166-2:US)* ISO 3166-2:CA *(https://en.wikipedia.org/wiki/ISO_3166-2:CA)*
postalCode	String	20	M	Postal code.
countryCode	String	2	M	Country code ("GB"). See ISO 3166-2 *(https://en.wikipedia.org/wiki/ISO_3166-2)*
paymentInfo	Object	N/A	M	Payment information object.
paymentType	String	50	M	Payment type ("CreditCard")
isDefault	Boolean	1	O	Indicates if a user has made this the default payment method. true / false | This attribute will only accept saveFlag=Y.
card	Object	N/A	M	Card information object.
cardNum	String	16	M	Card number
cardExpYear	String	4	M	Card expiry year.
cardExpMonth	String	2	M	Card expiry month.
cardCVVToken	String	20	M	SAKA token for decryption - card verification value.
cardHolder	String	50	M	Card holder.
cardNickName	String	20	O	Card nick-name.

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description

refPLOrderId	String	15	The unique Example OrderId reference for this transaction. For example: "201606020000013".
threeDSecureFlag	String	1	Determines if the 3DSecure process is required. "Y" = Required. "N" = Not required.
pspCode	String	10	Payment Service Provider Codes: • "SCTD" : SecureTrading • "ECP" : E-Com Processing
threeDSecureFlag is "Y"			
threeDSecure	Object	N/A	3DSecure information object.
xid	String	255	The unique identifier for the transaction, assigned by PSP.
enrolled	String	1	3DSecure enrolled Flag "Y" = Card is enrolled.
acsUrl	String	1024	The URL to be used in authenticating the card holder.
md	String	1024	The unique 3DSecure reference for this transaction. Note: This will not return when the pspCode is "ECP".
paReq	String	2048	The 3D authentication request. Note: This will not return when the pspCode is "ECP".
termURL	String	1024	The Example URL to be used for receiving 3DSecure result. Note: This will not return when the pspCode is "ECP".

Example: Case 1. Request with an existing PaymentInfoID

CheckTransaction HTTPS POST Request (JSON)

```json
{
  "orderId": "3a400f7c-42b2-43e3-ab81-b925dd3daf8e",
  "existingFlag": "Y",
  "paymentInfoId": "7eb5db22-c716-46f6-95d1-f54dc63fa3e1",
  "paymentInfo": {
    "paymentType": "CreditCard",
    "card": {
      "cardCvvToken": "1000000000002735"
    }
  },
  "transaction": {
    "applicationId": "cf9e3205-6cb3-42be-9df4-c2f418b74f2e",
    "applicationVersion": "0.4",
    "amount": 10,
    "currencyCode": "GBP",
    "extCasinoUserId": "281e77d9-48b4-4e59-a346-b64b3e5fa186",
    "extCasinoTransId": "1033a712-ff5b-4c8e-bb14-7d37552ef3dd",
    "extCasinoPaymentGatewayId": "5c8fac58-2c9f-4a44-87cd-6841648f48dc",
    "casinoId": "af1a90c6-f7ff-47bf-af01-86e077c1d85b",
    "categortNames": [
      "Poker",
      "Casino"
    ]
  },
  "gps": {
    "latitude": "37.554738",
    "longitude": "126.970680"
  },
  "accessCountryCode": "KR",
  "ipAddress": "112.223.50.121"
}
```

Encrypting Data and HTTPS Request [POST method]

```https
POST
http(s)://{host}/billing/323031363034323831363432
343563306334383539333864666644aa448bc71a97b5267514
c0b4c0ad7d54463f3a44f913391af5320bb01f188a11b501c
f5ffd075ffb1278f1c3a3d8ccc9b934b1b/transactions/
check

"Content-Type: application/x-www-form-urlencoded"
"Authorization: PLTOKEN {TOKEN}"

data=
3230313630343238313634323435323639313839656262323
1325be9680a1f756a284c8899774b1fb34ea880558f34f7f0
1b65ebb14cd1a48a958c23a823ba826ebff80ef83fd832879
114fa6101e8cb2001f51ed56574836927de54aced12591c95
ff9c8eb11aca4478f0d4bd7534549e2c580136014e3c92e3f
e12deaa7a7344449e065dcd52eb413b369c12db2e419c6de0
d08b43ace48b95e85fa73f2fbca061e33842d6593697c8546
0b7ba0cfaf04dee3efcc77468461ccd4d03435236f5753ee3
e4d5725ae73bdb96f0e1057534549e2c580136014e3c92e3f
e12deaa7a7344449e065dcd52eb413b369c12db2e419c6de0
d08b43ace48b95e85fa73f2fbca061e33842d6593697c8546
0b7ba0cfaf04dee3efcc77468461ccd4d03435236f5753ee3
e4d5725ae73bdb96f0e105
```

Successful *Encrypted* Response

```https
HTTPS STATUS CODE : 200 OK
```

```https
3230313530393234313530343332356662353634636132313
438818b824cf62c2d63feba0314c8b6cd3b328c2f02380787
406779be1607fa68363bf1fbac60a1b691de2f283ee2370f0
4f479287589b712576357c3256a2ba87f9c24d5948bb91dd1
3211ed9e452247603a75b5f5672f6718242e78baf48524182
e5f35ffa58874f3692f9ca9011d84d70e8f74dad430bf8a58
c8330fb0927e6048db45e3b18544a2d780e005903a668bbfb
a9fa7f0b5522ba0719c4e643c9fea11563fbd62d594c459d8
25f96ef53ed81aab2ec6a0657288c799e418e7e2d0451ca52
35a0d7dde3fe13f31cd6e7bd46470a58c2cb7372587b8e306
84fa98c2b6ddf3e06392bcbe41b80d107790ea58cc5b52115
6264ac9051b1b4aed931f9ba5b2bbe7e12b62de036f7cd1fb
328bfe80c8781a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

Note: In this case, the 3DSecure process **is required (threeDSecureFlag="Y" && enrolled="Y")**.

```https
HTTPS STATUS CODE : 200 OK
```

```json
{
  "data": {
    "refPLOrderId": "201606020000013",
```

```
    "threeDSecureFlag": "Y",
    "threeDSecure": {
      "xid": "K0w1N3gvMGZiSk9wVmNkc3JEVUg=",
      "enrolled": "Y",
      "acsUrl": "https://webapp.securetrading.net/acs/visa.cgi",
      "MD":
"UEZOVVBqeE5SRDQ4VFVSSVBtRnBURk00WkVaTFZHRm5jMGxP
U1ZacVJuUTBOMUU5UFR3dlRVUklQanhoWTNOVmNtdythSFIwY
0hNNkx5OTNaV0poY0hBdWMyVmpkWEpsZEhKaFpHbHVaeTV1Wl
hRdllXTnpMM1pwYzJFdVkyZHBQQzloWTNOVmNtdytQSEJoYmt
4bGJtZDBhRDR4Tmp3dmNHRnVUR1Z1WjNSb1BqeHRaWE56WVdk
bFNXUStVRUZTWlhFdE1UUTJORGcwTlRZeE9UY3hNeTB4TWpBN
U9USTBNVFEzUEM5dFpYTnpZV2RsU1dRK1BDOU5SRDQ4TDFOVV
BnPT06bWRXUW5CVFd3bEhEZC9xbXNBOFFmaDlSWUk5RjJ4clo
rZmt2Z2VKMmNqVHN3PQ==",
      "paReq":
"eJxVUttuwjAM/RXEOzhpCrTIRIIxbWwCobHxsLeSWlDRFkhT
YH+/pJRbnnx8O/Zx8HujicYLUqUmiVMqimhNjSQeNOfDLzq0u
N/1A7/T5WGPixb3WBh6Pvd7TYlVgsQj6SLZ5ZK3WdtDuELbS6
tNlBuJkTqMJjPJPWEbIdQQM9KTsWTPD+HixjzKSI7IqCgmhAq
h2pW50X8y8GybK8BSp3JjzL4PsErStDiq9upS11a7DE60Ak1F
mRowbtm4QHAlCPcJ56WzCktxTmL5yU58JtbH6dtvstiGp2U22
yrx8br8WQ8QXAbGkSHpMd5lXeY1WKcvRF+ECJUfo8zNJnm1Tw
1w7ziGj5FHD9oDaMrVdbkrQjrvdznZDKvtzcaYCnXjr4TjwrI
7L8J9m5d3p74yVlBf9O4qc87dHaqAY0isjjxgQUXhAIIrhfrE
UP8Laz39l39VT7et",
      "termURL": "https://billsvc.Example.com/web/result/threeds"
    }
  }
}
```


Note: In this case, the 3DSecure process is **not required (threeDSecureFlag="Y" && enrolled="N")**.

`HTTPS STATUS CODE : 200 OK`

json


```
{
  "data": {
    "refPLOrderId": "201606020000015",
    "threeDSecureFlag": "Y",
    "threeDSecure": {
      "xid": "TXU5Z3c2TTFKa2EzejJwc2hiSVI=",
      "enrolled": "N"
    }
  }
}
```


Note: In this case, the 3DSecure process is **not required (threeDSecureFlag="N")**.

`HTTPS STATUS CODE : 200 OK`

json


```
{
  "data": {
    "refPLOrderId": "201606020000019",
    "threeDSecureFlag": "N"
  }
}
```


Failed Response [HTTPS + JSON]

`HTTPS STATUS CODE : 404 Not Found`

```json
{
  "error": {
    "code": 906,
    "message": "cvv information does not exist.",
    "detail": null
  }
}
```

Example: Case 2. Request with New Payment Information

CheckTransaction HTTPS POST Request (JSON)

```json
{
  "orderId": "9e796f71-7752-44fe-ae2d-55e82f12c168",
  "existingFlag": "N",
  "saveFlag": "N",
  "billingInfo": {
    "firstName": "John",
    "lastName": "Doe",
    "dateOfBirth": "19870704",
    "phoneNumber": "1234567890",
    "emailAddress": "abc@xyz.com",
    "addressInfo": {
      "street": "18th Floore Prtlan House",
      "street2": "Bressenden Place",
      "city": "Victoria",
      "state": "London",
      "postalCode": "SW1E 5RS",
      "countryCode": "GB"
    }
  },
  "paymentInfo": {
    "paymentType": "CreditCard",
    "isDefault": true,
    "card": {
      "cardNum": "4370000000000111",
      "cardExpYear": "2018",
      "cardExpMonth": "08",
      "cardHolder": "John Doe",
      "cardNickName": "Card1",
      "cardCvvToken": "1000000000002741"
    }
  },
  "transaction": {
    "applicationId": "cf9e3205-6cb3-42be-9df4-c2f418b74f2e",
    "applicationVersion": "v0.1",
    "amount": 10,
    "currencyCode": "GBP",
    "extCasinoUserId": "45a06451-2d72-4228-8411-ef6997f8c4a7",
    "extCasinoTransId": "9399ab70-aee7-4ba3-bdbc-4e532f14b217",
    "extCasinoPaymentGatewayId": "8b056e5c-15c0-47e6-a25b-9cfb099e45f2",
    "casinoId": "af1a90c6-f7ff-47bf-af01-86e077c1d85b",
    "categortNames": [
      "Poker",
      "Casino"
    ],
  },
  "accessCountryCode": "KR",

```

```
  "ipAddress": "112.223.50.121"
}
```

Encrypting Data and HTTPS Request [POST method]

```https
POST
http(s)://{host}/billing/32303136303432383136343
234356330633438353933386466644aa448bc71a97b52675
14c0b4c0ad7d54463f3a44f913391af5320bb01f188a11b5
01cf5ffd075ffb1278f1c3a3d8ccc9b934b1b/transactions/
check

"Content-Type: application/x-www-form-urlencoded"
"Authorization: PLTOKEN {TOKEN}"

data=
32303136303432383136343234353236393138396562623231325be9680a1f756a284c8899774b1fb34ea880558f34f7
f01b65ebb14cd1a48a958c23a823ba826ebff80ef83fd832
879114fa6101e8cb2001f51ed56574836927de54aced1259
1c95ff9c8eb11aca4478f0d4bd7534549e2c580136014e3c
92e3fe12deaa7a7344449e065dcd52eb413b369c12db2e41
9c6de0d08b43ace48b95e85fa73f2fbca061e33842d65936
97c85460b7ba0cfaf04dee3efcc77468461ccd4d03435236
f5753ee3e4d5725ae73bdb96f0e1057534549e2c58013601
4e3c92e3fe12deaa7a7344449e065dcd52eb413b369c12db
2e419c6de0d08b43ace48b95e85fa73f2fbca061e33842d6
593697c85460b7ba0cfaf04dee3efcc77468461ccd4d0343
5236f5753ee3e4d5725ae73bdb96f0e105
```

Successful *Encrypted* Response [HTTPS]

`HTTPS STATUS CODE : 200 OK`

```https
32303135303932343135303433323566623536346361323133438818b824cf62c2d63feba0314c8b6cd3b328c2f02380787
406779be1607fa68363bf1fbac60a1b691de2f283ee2370f0
4f479287589b712576357c3256a2ba87f9c24d5948bb91dd1
3211ed9e452247603a75b5f5672f6718242e78baf48524182
e5f35ffa58874f3692f9ca9011d84d70e8f74dad430bf8a58
c8330fb0927e6048db45e3b18544a2d780e005903a668bbfb
a9fa7f0b5522ba0719c4e643c9fea11563fbd62d594c459d8
25f96ef53ed81aab2ec6a0657288c799e418e7e2d0451ca52
35a0d7dde3fe13f31cd6e7bd46470a58c2cb7372587b8e306
84fa98c2b6ddf3e06392bcbe41b80d107790ea58cc5b52115
6264ac9051b1b4aed931f9ba5b2bbe7e12b62de036f7cd1fb
328bfe80c8781a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response [HTTPS + JSON]

Note: In this case, the 3DSecure process **is required** **(threeDSecureFlag="Y" && enrolled="Y")**.

`HTTPS STATUS CODE : 200 OK`

```json
{
  "data": {
    "refPLOrderId": "201606020000018",
    "threeDSecureFlag": "Y",
    "threeDSecure": {
```

```
      "xid": "TXU5Z3c2TTFKa2EzejJwc2hiSVI=",
      "enrolled": "Y",
      "acsUrl": "https://webapp.securetrading.net/acs/visa.cgi",
      "md":
"UEZOVVBqeE5SRDQ4VFVSSVBqWnZNRUZSZEhsTVpIZHViMVZa
ZGsxdFQwOVpNVkU5UFR3dlRVUklQanhoWTNOVmNtdythSFIwY
0hNNkx5OTNaV0poY0hBdWMyVmpkWEpsZEhKaFpHbHVaeTV1Wl
hRdllXTnpMM1pwYzJFdVkyZHBQQzloWTNOVmNtdytQSEJoYmt
4bGJtZDBhRDR4Tmp3dmNHRnVUR1Z1WjNSb1BqeHRaWE56WVdk
bFNXUStVRUZTWlhFdE1UUTJORGcwT0RnNU9EVTRNaTAzTVRNN
E16UXhNRGM4TDIxbGMzTmhaMlZKWkQ0OEwwMUVQand2VTFRKz
ptZFdRbkJUV3dsSERkL3Ftc0E4UWZoOWRqY3BER1cwWERzZGR
oRVVoak5rRVU9",
      "paReq":
"eJxVUm1PwjAQ/iuG7+zWboxKjiYgGtFoiExj/Da6i5sZA7oO
0F9vO8ZbP91zb8/dc8U400STOalak8QXqqrkm27ydNiZjd5o0
2VhFIpQiFvRE7zbZ4EIQub3OxKbuMQt6SpflZJ5vscRjtC20i
pLSiMxUZvx9FUyHoS9CKGFuCQ9nUj/+iEc3FgmS5JjMipJCaF
BqFZ1afSvFNy2OQKsdSEzY9YDgEVeFNVWeYtDnadWS9jRAjRV
dWHAuF3TCsGVIJwnnNXOqizFPk9l/Pne+woUj+OH54Tf/9HP0
07xLJ9/TIcILgPTxJDkPov8yOc3fjTgYsAEQuPHZOlmk6zZpw
W4dhyjy8ilB63+mkp1XO6IkPbrVUk2w2p7sjGlSp34G+Eadud
FOG9z9+jUV8YKGgb9s8qMMXeHJuAYcqsjE75oKBxAcKXQnhja
b2Gtq+/yD8r8t2E=",
      "termURL": "https://billsvc.Example.com/web/result/threeds"
    }
  }
}
```


Note: In this case, the 3DSecure process is **not required** **(threeDSecureFlag="Y" && enrolled="N")**.

`HTTPS STATUS CODE : 200 OK`

json


```
{
  "data": {
    "refPLOrderId": "201606020000019",
    "threeDSecureFlag": "Y",
    "threeDSecure": {
      "xid": "TXU5Z3c2TTFKa2EzejJwc2hiSVI=",
      "enrolled": "N"
    }
  }
}
```


Note: In this case, the 3DSecure process is **not required** **(threeDSecureFlag="N")**.

`HTTPS STATUS CODE : 200 OK`

json


```
{
  "data": {
    "refPLOrderId": "201606020000019",
        "threeDSecureFlag": "N"
    }
}
```


Failed Response [HTTPS + JSON]

`HTTPS STATUS CODE : 404 Not Found`

```json
{
  "error": {
    "code": 906,
    "message": "cvv information does not exist.",
    "detail": null
  }
}
```

POST Create Billing Transaction with New Payment Information

(CreateTransactionWithNewPaymentInfo)

Description

A server can invoke this action to request a billing transaction with new payment information.

Endpoint url

/billing/{uuid}/transactions

HTTPS Method

POST

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	36	M	User's unique identification
orderId	String	36	M	Unique OrderId for a transaction by the client side.
refPLOrderId	String	15	M	Must be the value returned in the refPLOrderId field of the "Check Billing Transaction" API Response.
ipAddress	String	50	M	User's IP Address
coreData	Object	N/A	M	Core Object
additionalIdentifier	String	N/A	M	Can be "CUSTOM" or "PRESET"
promoCodes	Array (Object)	N/A	O	Array of the Promotion Object. Is set by MC as part of pre-check.
promoCode	String	N/A	M	A promotion code.
bonusAmount	Double	8	M	A bonus amount.
promotionalBonus	Array (Object)	N/A	O	An array of the Promotion Object. Is set by MC as part of pre-check.
promotionalAppId	String	N/A	M	An Application ID as per the core database. Should typically come from core database.
bonusAmount	Double	8	M	A bonus amount.
ipAddress	String	50	M	User's IP Address.
threeDSecureFlag	String	1	M	Determines if this is a 3DSecure transaction. "Y" = A 3DSecure transaction. "N" = Not a 3DSecure transaction.
threeDSecureFlag = "Y"				
threeDSecure	Object	N/A	O	3DSecure information object.
md	String	1024	O	Must be the value returned in the md field returned from Example in the 3DScecure Process.
paRes	String	65536	O	Must be the value returned in the md field returned from Example in the 3DScecure Process.
xid	String	255	M	Must be the value returned in the md field returned from Example in the 3DScecure Process.

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
uuid	String	50	User's Unique ID.

JSON Property	Type	Size	Description
paymentInfoId	String	36	Payment Information ID. If saveflag = "Y", paymentInfoId will be returned.
transaction	Object	N/A	Transaction object.
applicationId	String	36	Unique ID for a game.
amount	Double	8	Payment amount.
currencyCode	String	3	Payment currency code: ISO 4217 *(https://en.wikipedia.org/wiki/ISO_4217)*
transactionOrderId	String	36	Corresponds to a Deposit Transaction OrderID.
transactionId	String	36	Unique ID of Example.
status	String	10	Status of transaction SUCCESS.
transactionDate	String	20	Date and time of transaction.
extCasinoUserId	String	N/A	Corresponds to a casino's user ID.
extCasinoTransId	String	N/A	Corresponds to a casino's transaction ID.
extCasinoPaymentGatewayId	String	N/A	Corresponds to a casino's wallet ID.

HTTPS Request example: CreateTransaction [POST method]

```json
{
  "orderId": "bebd843d-1add-443d-8586-b066f081d757",
  "refPLOrderId": "201606020000023",
  "threeDSecureFlag": "Y",
  "threeDSecure": {
    "md": "UEZOVVBqeE5SRDQ4VF…ZoOWIyMGhPTldHT1ZlRnFLUkZRS0FwZ009",
    "paRes": "eJztWMmyq7iynfMVFVVDxyl6DBU+…2meb0lv566n29n3z6B/wsHIW/A",
    "xid" : "TXU5Z3c2TTFKa2EzejJwc2hiSVI=",
  },
  "coreData": {
    "additionalIdentifier": "CUSTOM",
    "applicationId": "cf9e3205-6cb3-42be-9df4-c2f418b74f2e"
  },
  "ipAddress": "112.223.50.121"
}
/*  Please note that the "md" and "paRes" fields typically contain values
    far longer than displayed in the example above and therefore have been
    abbreviated using "…", for your convenience.
*/
```

Encrypting Data and HTTPS Request [POST method]

```https
POST
http(s)://{host}/billing/32303135303932343135303
63231336263383630333332313836e14946e58b0485ae4ab
000/transactions

"Content-Type: application/x-www-form-urlencoded"
"Authorization: PLTOKEN {TOKEN}"

data=
32303135303932343135303632313362633836303333323 13
836e14946e58b0485ae4ab000557942caa75e84708a6123ac
21a9945845a07d557a6bee23a9b01db139e3d5895bae32047
```

```
6cc665ac7b5864baeab3a806570b5e4f9b2ba8a4085764391
3f93e6534753ab08be5815a0d170d0f679a3c3d838ae885e8
0f01f40bd45a0d506858e6c97ca0e55d0aa565c79b3a5e34a
b81c4e7c9b754557231be0760e8041eb7cbba8a0e9db51a93
c99917e39cf1cda93412b1c20795ce171e9b9e316006cb783
0085b0434eb2c90a12d749f975c09f3fe7ccc11242c7452ed
b5325baa9a852298ab059d40e09a3b653e64c953adddf04f6
761747d66827820035c58452c2fa9c72af00a7caa79a1d39c
a467acc7dd05b58ed2f36ca02f20fd4523fcc84361ff7f7cd
81985ff283f71cbd4cae958e6055a127cba408b23d3bee884
e04a986eb94a943442f428e4daa207af947d3a03faa77f47b
2ca7f06382af36567e3cb6ead4f19ffdfc850111971e78b68
453abea1aa650a2eacf00505dbc84c1b6aa2b2c1f9814bb34
997e8e13906810ea92533c823df4a2ebfd4fcf7683328b452
a360736fd65d5d1ddad22466078bd74aa9705e7b58e58a846
ccb46b9fa9ea8106a6b520f596bc44407541d8bfbc54f9579
7895e758076864502aef033b7223db03ad35d0d1934d67e18
dd58a0fba2da2a323bae7fc971829eb68edfc69e0ae873cd1
df8099509
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

https

```
32303135303932343135303433323566623536346361323 13
438818b824cf62c2d63feba0314c8b6cd3b328c2f02380787
406779be1607fa68363bf1fbac60a1b691de2f283ee2370f0
4f479287589b712576357c3256a2ba87f9c24d5948bb91dd1
3211ed9e452247603a75b5f5672f6718242e78baf48524182
e5f35ffa58874f3692f9ca9011d84d70e8f74dad430bf8a58
c8330fb0927e6048db45e3b18544a2d780e005903a668bbfb
a9fa7f0b5522ba0719c4e643c9fea11563fbd62d594c459d8
25f96ef53ed81aab2ec6a0657288c799e418e7e2d0451ca52
35a0d7dde3fe13f31cd6e7bd46470a58c2cb7372587b8e306
84fa98c2b6ddf3e06392bcbe41b80d107790ea58cc5b52115
6264ac9051b1b4aed931f9ba5b2bbe7e12b62de036f7cd1fb
328bfe80c8781a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK

json


```
{
  "data": {
    "uuid": "dc79042f-321e-42df-9e12-25fc2a4a8f1a",
    "transaction": {
      "applicationId": "cf9e3205-6cb3-42be-9df4-c2f418b74f2e",
      "amount": 10,
      "currencyCode": "GBP",
      "transactionOrderId": "cf9e3205-6cb3-42be-9df4-c2f418b74f2e",
      "transactionId": "201606020000024",
      "status": "SUCCESS",
      "transactionDate": "02/06/2016 06:48:10",
      "extCasinoUserId": "319ddbd2-1a6b-4064-a5a5-e6ef920be2cf",
      "extCasinoTransId": "9cdfb42a-ba6f-4242-9e0d-f295c99003b7",
      "extCasinoPaymentGatewayId": "c02b1adf-4efd-4eff-963e-3a025a4a5231"
    }
  }
}
```

Failed Response [HTTPS + JSON]

HTTPS STATUS CODE : 500 Internal Server Error

```json
{
  "error": {
    "code": 981,
    "message": "PSP processing failed",
    "detail" : "Invalid Card Number"
  }
}
```

POST Create Billing Transaction with PaymentInfoID (CreateTransactionWithPaymentInfoID)

Description

Create a billing transaction with a PaymentInfoID. A server can invoke this action to request a billing transaction with saved payment information.

Endpoint url

/billing/{uuid}/paymentinfo/{paymentinfoid}/transactions

HTTPS Method

POST

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	36	M	Encrypted user's unique identification.
paymentInfoId	String	36	M	User's unique payment information ID.
orderId	String	36	M	Unique OrderId for a client-side transaction.
refPLOrderId	String	15	M	Must be the value returned in the refPLOrderId field of the "Check Billing Transaction" API Response.
ipAddress	String	50	M	User's IP Address
coreData	Object	N/A	M	Core object.
additionalIdentifier	String	N/A	M	Can be "CUSTOM" or "PRESET".
promoCodes	Array (Object)	N/A	O	Array of the Promotion Object. Is set by MC as part of pre-check.
promoCode	String	N/A	M	A promotion code.
bonusAmount	Double	8	M	A bonus amount.
promotionalBonus	Array (Object)	N/A	O	An array of the Promotion Object. Is set by MC as part of pre-check.
promotionalAppId	String	N/A	M	An Application ID as per the core database. Should typically come from core database.
bonusAmount	Double	8	M	A bonus amount.
ipAddress	String	50	M	User's IP Address.
threeDSecureFlag	String	1	M	Determines if the transaction is 3DSecure. "Y" = A 3DSecure transaction. "N" = Not a 3DSecure transaction.
threeDSecureFlag = "Y"				
threeDSecure	Object	N/A	O	3DSecure information object.
md	String	1024	O	Must be the value returned in the md field returned from Example in the 3DScecure Process.
paRes	String	65536	O	Must be the value returned in the md field returned from Example in the 3DScecure Process.
xid	String	255	M	Must be the value returned in the md field returned from Example in the 3DScecure Process.

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description

uuid	String	50	User's Unique ID.
paymentInfoId	String	36	Payment Information ID. If saveflag = "Y", paymentInfoId will be returned.
transaction	Object	N/A	Transaction object.
applicationId	String	36	Unique ID for a game.
amount	Double	8	Payment amount.
currencyCode	String	3	Payment currency code: ISO 4217 *(https://en.wikipedia.org/wiki/ISO_4217)*
transactionOrderId	String	36	Corresponds to a Deposit Transaction OrderId.
transactionId	String	36	Unique ID of Example.
status	String	10	Status of transaction, for example: SUCCESS.
transactionDate	String	20	Date and time of transaction.
extCasinoUserId	String	N/A	Corresponds to a casino's user ID.
extCasinoTransId	String	N/A	Corresponds to a casino's transaction ID.
extCasinoPaymentGatewayId	String	N/A	Corresponds to a casino's wallet ID.

HTTPS Request example: CreateTransactionByPMID [POST method]

```json
{
  "orderId": "d4127f51-3078-4d78-ba24-bb0c9d3c16c2",
  "refOrderId": "201606020000027",
  "threeDSecureFlag": "N",
  "coreData": {
    "additionalIdentifier": "CUSTOM",
    "applicationId": "cf9e3205-6cb3-42be-9df4-c2f418b74f2e",
    "promoCodes": [
      {
        "promoCode": "cb108584-333c-4db9-abb8-7726554fce13",
        "bonusAmount": 10
      },
      {
        "promoCode": "e7c58cc2-0579-44ca-8e3f-d391ff8b43cf",
        "bonusAmount": 10
      },
      {
        "promoCode": "5d1ab6be-8e99-49d0-a15d-c863f59a065f",
        "bonusAmount": 10
      }
    ]
  },
  "ipAddress": "112.223.50.121"
}
```

Encrypting Data and HTTPS Request [POST method]

```https
POST
http(s)://{host}/billing/3230313530393234313530363231336
6323133626338363033333231383 6e14946e58b0485ae4ab
000/paymentinfo/3230313530393234313530363231336
6338363033333231383 6e14946e58b0485ae4ab000/transactions

"Content-Type: application/x-www-form-urlencoded"
```

```
"Authorization: PLTOKEN {TOKEN}"

data=
32303135303932343135303632313362633836303333323138363e14946e58b0485ae4ab000557942caa75e84708a6123ac21a9945845a07d557a6bee23a9b01db139e3d5895bae320476cc665ac7b5864baeab3a806570b5e4f9b2ba8a40857643913f93e6534753ab08be5815a0d170d0f679a3c3d838ae885e80f01f40bd45a0d506858e6c97ca0e55d0aa565c79b3a5e34ab81c4e7c9b754557231be0760e8041eb7cbba8a0e9db51a93c99917e39cf1cda93412b1c20795ce171e9b9e316006cb7830085b0434eb2c90a12d749f975c09f3fe7ccc11242c7452edb5325baa9a852298ab059d40e09a3b653e64c953adddf04f6761747d66827820035c58452c2fa9c72af00a7caa79a1d39ca467acc7dd05b58ed2f36ca02f20fd4523fcc84361ff7f7cd81985ff283f71cbd4cae958e6055a127cba408b23d3bee884e04a986eb94a943442f428e4daa207af947d3a03faa77f47b2ca7f06382af36567e3cb6ead4f19ffdfc850111971e78b68453abea1aa650a2eacf00505dbc84c1b6aa2b2c1f9814bb34997e8e13906810ea92533c823df4a2ebfd4fcf7683328b452a360736fd65d5d1ddad22466078bd74aa9705e7b58e58a846ccb46b9fa9ea8106a6b520f596bc44407541d8bfbc54f95797895e758076864502aef033b7223db03ad35d0d1934d67e18dd58a0fba2da2a323bae7fc971829eb68edfc69e0ae873cd1df8099509
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

```https
32303135303932343135303433323566623536346361323138b824cf62c2d63feba0314c8b6cd3b328c2f02380787406779be1607fa68363bf1fbac60a1b691de2f283ee2370f04f479287589b712576357c3256a2ba87f9c24d5948bb91dd13211ed9e452247603a75b5f5672f6718242e78baf48524182e5f35ffa58874f3692f9ca9011d84d70e8f74dad430bf8a58c8330fb0927e6048db45e3b18544a2d780e005903a668bbfba9fa7f0b5522ba0719c4e643c9fea11563fbd62d594c459d825f96ef53ed81aab2ec6a0657288c799e418e7e2d0451ca5235a0d7dde3fe13f31cd6e7bd46470a58c2cb7372587b8e30684fa98c2b6ddf3e06392bcbe41b80d107790ea58cc5b521156264ac9051b1b4aed931f9ba5b2bbe7e12b62de036f7cd1fb328bfe80c8781a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK

```json
{
  "data": {
    "uuid": "dc79042f-321e-42df-9e12-25fc2a4a8f1a",
    "paymentInfoId": "8bae8471-0785-4c39-b016-086f8d54d58a",
    "transaction": {
      "applicationId": "cf9e3205-6cb3-42be-9df4-c2f418b74f2e",
      "amount": 10,
      "currencyCode": "GBP",
      "transactionOrderId": "cf9e3205-6cb3-42be-9df4-c2f418b74f2e",
      "transactionId": "201606020000028",
      "status": "SUCCESS",
      "transactionDate": "02/06/2016 07:04:20",
```

```
      "extCasinoUserId": "18477d20-8737-470d-af99-038236c87487",
      "extCasinoTransId": "5d6acd38-72b2-4348-adbb-d6561aaa4d4f",
      "extCasinoPaymentGatewayId": "9bcdac44-7a4d-4f03-8fbc-7520aca91a02"
    }
  }
}
```


Failed Response

HTTPS STATUS CODE : 500 Internal Server Error

json


```
{
  "error": {
    "code": 981,
    "message": "PSP processing failed",
    "detail" : "Invalid Card Number"
  }
}
```

GET Get Billing Transaction (GetTransactions)

Description

A server can invoke this action to get a billing transaction list.

Endpoint url

/billing/{uuid}/transactions

HTTPS Method

GET

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	200	M	Encrypted user's unique ID.
pageNo	Integer	4	M	Start number of page.
pageSize	Integer	4	M	Row number per page.
fromDate	String	8	M	Start date of transaction. Example: (YYYYMMDD) as in "20151231".
toDate	String	8	M	End date of transaction. Example: (YYYYMMDD) as in "20151231".

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
uuid	String	200	User's unique ID.
totalCount	Integer	4	Total count of a transaction.
transactions	Array (Object)	N/A	Array of the transaction object.
transactionId	String	36	Unique transaction ID.
paymentDate	String	20	Date and time of payment.
applicationId	String	15	Related game code of payment.
amount	Double	8	Payment amount.
currencyCode	String	3	Payment currency code. ISO 4217 *(https://en.wikipedia.org/wiki/ISO_4217)*
paymentType	String	50	Payment Type: "CreditCard".

HTTPS Request example: GetTransactions [GET method]

```
https
/billing/89396B99-5A57-4ECA-96C9-60ED109F8C8A/transactions?
pageNo=1&
pageSize=10&
fromDate=20150101&
toDate=20150831
```

Encrypting Data and HTTPS Request [GET method]

```
https
GET
https://{host}/billing/32303135303932343135303934 3
26234663364343639316361349 91a219bba46116e3deecfe4
85f650a558f861cafb503295930eccc7ce11b99f57524ef67
713f45582a5eb07e9fe8f1d9e19c933/transactions?
```

```
data=
3230313530393232313631373330633062613738323263303
839e61251256b1bd3583b2f02642a2e678debda2f67a685d3
e56e034f8

"Authorization: PLTOKEN {TOKEN}"
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

```https
3230313530393234313530343332356662353634636132313
438818b824cf62c2d63feba0314c8b6cd3b328c2f02380787
406779be1607fa68363bf1fbac60a1b691de2f283ee2370f0
4f479287589b712576357c3256a2ba87f9c24d5948bb91dd1
3211ed9e452247603a75b5f5672f6718242e78baf48524182
e5f35ffa58874f3692f9ca9011d84d70e8f74dad430bf8a58
c8330fb0927e6048db45e3b18544a2d780e005903a668bbfb
a9fa7f0b5522ba0719c4e643c9fea11563fbd62d594c459d8
25f96ef53ed81aab2ec6a0657288c799e418e7e2d0451ca52
35a0d7dde3fe13f31cd6e7bd46470a58c2cb7372587b8e306
84fa98c2b6ddf3e06392bcbe41b80d107790ea58cc5b52115
6264ac9051b1b4aed931f9ba5b2bbe7e12b62de036f7cd1fb
328bfe80c8781a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK


```json
{
  "data": {
    "uuid": "2a50a2e4-a93a-4823-9674-c63aba792325",
    "totalCount": 3,
    "transactions": [
      {
        "applicationId": "00f378f3-f467-4a39-81ea-ac8b0d073eea",
        "amount": 10,
        "currencyCode": "GBP",
        "transactionId": "201604280000070",
        "paymentType": "CREDITCARD",
        "transactionDate": "28/04/2016 16:52:00"
      },
      {
        "applicationId": "00f378f3-f467-4a39-81ea-ac8b0d073eea",
        "amount": 10,
        "currencyCode": "GBP",
        "transactionId": "201604280000066",
        "paymentType": "CREDITCARD",
        "transactionDate": "28/04/2016 16:51:55"
      },
      {
        "applicationId": "00f378f3-f467-4a39-81ea-ac8b0d073eea",
        "amount": 10,
        "currencyCode": "GBP",
        "transactionId": "201604280000062",
        "paymentType": "CREDITCARD",
        "transactionDate": "28/04/2016 16:51:29"
      }
    ]
```

```
  }
}

```

Failed Response

HTTPS STATUS CODE : 404 Not Found

```json
{
  "error": {
    "code": 911,
    "message": "This account (89396B99-5A57-4ECA-96C9-60ED109F8C8A) does not have transactions.",
    "detail": null
  }
}
```

GET Get Transaction Count (GetTransactionCount)

Description

A server can invoke this action to get a transaction count.

Endpoint url

/billing/transactions/{transactionDate}/count

HTTPS Method

GET

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
transactionDate	String	N/A	M	Encrypted transaction date (YYYYMMDD), as in "20160106".
countryCode	String	2	M	Country code ("GB"): ISO 3166-2 *(https://en.wikipedia.org/wiki/ISO_3166-2)*
status	String	10	M	Status of transaction: SUCCESS, FAILED, REFUNDED, REVERSED.

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
totalCount	Integer	4	Count of matched transactions.

HTTPS Request example: GetTransactionCount [GET method]

```
https
/billing/transactions/20160106/count?
countryCode=GB&
status=SUCCESS
```

Encrypting Data and HTTPS Request [GET method]

```
https
GET
https://{host}/billing/transactions/32303135303932
343135303934326234663364343639316361349 1a219bba4
6116e3deecfe485f650a558f861cafb503295930eccc7ce11
b99f57524ef67713f45582a5eb07e9fe8f1d9e19c933?

data=
323031353039323431353039343262346633643436393163613491a219bba46116e3deecfe485f650a558f861cafb5032
95930eccc7ce11b99f57524ef67713f45582a5eb07e9fe8f1
d9e19c93

"Authorization: PLTOKEN {TOKEN}"
```

Successful *Encrypted* Response [HTTPS]

HTTPS STATUS CODE : 200 OK

```
https
323031353039323431353034333235666235363463613231 3
438818b824cf62c2d63feba0314c8b6cd3b328c2f02380787
406779be1607fa68363bf1fbac60a1b691de2f283ee2370f0
4f479287589b712576357c3256a2ba87f9c24d5948bb91dd1
3211ed9e452247603a75b5f5672f6718242e78baf48524182
e5f35ffa58874f3692f9ca9011d84d70e8f74dad430bf8a58
```

```
c8330fb0927e6048db45e3b18544a2d780e005903a668bbfb
a9fa7f0b5522ba0719c4e643c9fea11563fbd62d594c459d8
25f96ef53ed81aab2ec6a0657288c799e418e7e2d0451ca52
35a0d7dde3fe13f31cd6e7bd46470a58c2cb7372587b8e306
84fa98c2b6ddf3e06392bcbe41b80d107790ea58cc5b52115
6264ac9051b1b4aed931f9ba5b2bbe7e12b62de036f7cd1fb
328bfe80c8781a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK

json

```json
{
  "data": {
    "totalCount": 10
  }
}
```

Failed Response

HTTPS STATUS CODE : 500 Internal Server Error

json

```json
{
  "error": {
    "code": 999,
    "message": "Internal Error.",
    "detail": null
  }
}
```

GET Get Transaction History List (GetTransactionHistory)

Description

A server can invoke this action to get a transaction history list.

Endpoint url

/billing/transactions/{transactionDate}/history

HTTPS Method

GET

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
transactionDate	String	N/A	M	Encrypted transaction date (YYYYMMDD), as in "20160106".
countryCode	String	2	M	Country code ("GB"): ISO 3166-2 *(https://en.wikipedia.org/wiki/ISO_3166-2)*
status	String	10	M	Status of transaction: SUCCESS, FAILED, REFUNDED, REVERSED.

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
totalCount	Integer	4	Total count of a transaction.
transactions	Array	N/A	Array of the transaction object.
transactionID	String	50	Unique ID of Example.
orderID	String	50	Unique ID for a client-side transaction.
uuid	String	50	User's unique ID.
paymentInfoId	String	50	Payment Information ID.
applicationId	String	15	Related game code of a transaction.
amount	Double	8	Payment amount.
currencyCode	String	3	Payment currency code: ISO 4217 *(https://en.wikipedia.org/wiki/ISO_4217)*
status	String	10	Status of transaction: SUCCESS, FAILED, REFUNDED, REVERSED.
comment	String	200	Note of a transaction.
ipAddress	String	50	IP Address.
transactionDate	String	20	Date and time of transaction.

HTTPS Request example: GetTransactionHistory [GET method]

```
https
/billing/transactions/20160106/history?
countryCode=GB&
status=SUCCESS
```

Encrypting Data and HTTPS Request [GET method]

```
https
GET
https://{host}/billing/transactions/3230313530393
2343135303934326234663364343639316361349 91a219bb
a46116e3deecfe485f650a558f861cafb503295930eccc7c
e11b99f57524ef67713f45582a5eb07e9fe8f1d9e19c933/
```

```
history?

data=
3230313530393234313530393432623466336434363931636134991a219bba46116e3deecfe485f650a558f861cafb503295930eccc7ce11b99f57524ef67713f45582a5eb07e9fe8f1d9e19c93

"Authorization: PLTOKEN {TOKEN}"
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

```https
3230313530393234313530343332356662353634636132313438818b824cf62c2d63feba0314c8b6cd3b328c2f02380787406779be1607fa68363bf1fbac60a1b691de2f283ee2370f04f479287589b712576357c3256a2ba87f9c24d5948bb91dd13211ed9e452247603a75b5f5672f6718242e78baf48524182e5f35ffa58874f3692f9ca9011d84d70e8f74dad430bf8a58c8330fb0927e6048db45e3b18544a2d780e005903a668bbfba9fa7f0b5522ba0719c4e643c9fea11563fbd62d594c459d825f96ef53ed81aab2ec6a0657288c799e418e7e2d0451ca5235a0d7dde3fe13f31cd6e7bd46470a58c2cb7372587b8e30684fa98c2b6ddf3e06392bcbe41b80d107790ea58cc5b521156264ac9051b1b4aed931f9ba5b2bbe7e12b62de036f7cd1fb328bfe80c8781a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK

```json
{
  "data": {
    "totalCount": 3,
    "transactions": [
      {
        "transactionID": "201508280404206",
        "orderID": "cedad85e-fdea-40e1-baeb-2384786aaeb0",
        "uuid": "79d97b31-f48a-4dd4-a9be-1ea9cd4998d4",
        "paymentInfoId": "995d9b68-3eae-4fd9-b166-1b5df602b640",
        "applicationId": "ABC",
        "amount": 10,
        "currencyCode": "GBP",
        "status": "SUCCESS",
        "comment": "Your transaction was approved by Paypal",
        "ipAddress": "127.0.0.1",
        "transactionDate": "20/12/2015 15:10:10"
      },
      {
        "transactionID": "201508280404207",
        "orderID": "d3b173a4-afc5-4209-8424-fd8c237c7f7c",
        "uuid": "79d97b31-f48a-4dd4-a9be-1ea9cd4998d4",
        "paymentInfoId": "c9455a07-59a6-418c-90f8-a98398379d84",
        "applicationId": "ABC",
        "amount": 30,
        "currencyCode": "GBP",
        "status": "FAIL",
        "comment": "Invalid credit card information. Please re-enter. (Invalid account number)",
        "ipAddress": "127.0.0.1",
```

```
        "transactionDate": "20/12/2015 08:10:10"
      },
      {
        "transactionID": "201508280404208",
        "orderID": "745125db-5bf6-457c-a59e-b8968abf3ea8",
        "uuid": "6633d40c-b253-482f-99b3-41385f80fefb",
        "paymentInfoId": "22a1787f-d492-4369-8e08-aaa3e8dd8478",
        "applicationId": "ABC",
        "amount": 50,
        "currencyCode": "GBP",
        "status": "SUCCESS",
        "comment": "Your transaction was approved by Paypal",
        "ipAddress": "127.0.0.1",
        "transactionDate": "20/12/2015 18:10:10"
      }
    ]
  }
}

```


Failed Response [HTTPS + JSON]

HTTPS STATUS CODE : 500 Internal Server Error

json


```
{
  "error": {
    "code": 999,
    "message": "Internal Error.",
    "detail": null
  }
}
```

POST Payout Transaction (Payout)

Description

A server can invoke this action to create a payout transaction.

Endpoint url

/billing/{uuid}/payout

HTTPS Method

POST

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	36	M	User's unique identification.
orderId	String	36	M	Unique OrderId for a transaction by the client side.
paymentInfoId	String	36	O	Payment Information ID.
transaction	Object	N/A	M	Transaction Object.
applicationId	String	36	O	Unique ID for Casino (application).
applicationVersion	String	10	O	Casino (application) Version.
amount	Double	8	M	Requested amount.
currencyCode	String	3	M	Currency code.
extCasinoUserId	String	N/A	O	Corresponds to a casino's user id.
extCasinoTransId	String	N/A	O	Corresponds to a casino's transaction id.
extCasinoPaymentGatewayId	String	N/A	O	Corresponds to a casino's wallet id.

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
uuid	String	36	User's unique Identification.
paymentInfoId	String	36	Payment Information ID.
transaction	Object (Array)	N/A	Array of the transaction object.
transactionID	String	50	Unique ID of Example.
transactionOrderId	String	36	Unique ID for a Payout Transaction.
currencyCode	String	3	Payment currency code: ISO 4217 *(https://en.wikipedia.org/wiki/ISO_4217)*
amount	Double	8	Payment amount.
pspCode	String	10	Payment Service Provider Codes: • "SCTD" : SecureTrading • "ECP" : E-Com Processing
pspTid	String	50	Unique ID from a Payment Service Provider - "Claim by user".

JSON Property	Type	Size	Description
region	String	40	Region for a user's payment information: • "Domestic" • "Intra Europe EEA" • "Intra Europe (Non-EEA)" • "International"

HTTPS Request example: Payout [POST method]

```json
{
  "uuid":"00d8f6a8-edbe-4eee-9881-b347b740c852",
  "orderId":"9ef2eb6a-e265-4903-9f82-81f35ac6bc59",
  "transaction":{
    "applicationId":"0e358e8e-4f2f-4c67-affb-ccbd5847eb14",
    "applicationVersion":"0.1",
    "amount":10.0,
    "currencyCode":"GBP",
    "extCasinoUserId":"34b72420-7ca7-4748-a966-25d058e303c7",
    "extCasinoTransId":"d0cbed47-aa1c-45ce-b313-9eb491345749",
    "extCasinoPaymentGatewayId":"06874048-52c7-400a-9718-d9f7653523a0"
  },
  "ipAddress":"203.170.118.12"
}
```

Encrypting Data and HTTPS Request [POST method]

```https
POST http(s)://{host}/billing/{uuid}/payout
"Content-Type: application/x-www-form-urlencoded"
"Authorization: PLTOKEN {TOKEN}"
data=
32303136303730383137333834396332383833336638323266326e053ecdf27192b78f1aa4c4e176381395ae5051f12a9d6
c792ed29f7cd81e8682b28f6350650f82906e866f64ea529d44746b78eb57a266a11d70189e49ab645505c266e6db3c530
9f0a8d8d3a0106267c31bd792b18f224b2da5cf95b9d9b5965db91c2959449e5cb21a967ebec0b09704acc8c673fc0c4535
9b769fec78690af0a0398d6e4a53b55f34f87e383778742bb68a7bc149b87d905652305a70dc249b74d33316629d5dc9fce
e98a46e917666ed9787d411e1ab4826e5c273e1b6ce1f0cba86ddb2e007035fe5597bfba7b23885563f628af36b7d2c6703
48717a59bb659285109de37f7f63ed67e9066d7e16ffdf4045acde499968baa2ed25cbd7594c4e98e6aaac384a1b4a7f69b
420a546b708398b145b050b4c2bcd2db4525b70b8b8e807fb266bffbaa9ae66ae293e2245391756b86ff0b0bc0427cbba3
ee29e0d64cdd01d3db243c3802fe00ad8385acdcf16c923a8cdb00c87bb992e6baf46508ce744fd35d1b1f981af29f46734e
c48ed4d1cfea37d272f11a26bc929d993324766945254a880832d33390c74812958bef359d8bbabf734a48bbb63fc1ccdd0
82cd872514cb505e5cb16cbd9c3aba4e21ea8d9475c62a0b1b6a532afdc96db2c0d10c60bb6f0c12b15c8e1475600e013c
a4d0ff80a01a8c8019c38c8d6ecc7b453517e5a4ce5dc4983743859b864822b7a0db79ea248599b6f52c585dd2e31e5caf1
c1b134ed42809eccfceb98031583f7a31cbd52fffd8afc8d
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

```https
32303135303932343135303433323566623536346361323
438818b824cf62c2d63feba0314c8b6cd3b328c2f02380787
406779be1607fa68363bf1fbac60a1b691de2f283ee2370f0
4f479287589b712576357c3256a2ba87f9c24d5948bb91dd1
3211ed9e452247603a75b5f5672f6718242e78baf48524182
e5f35ffa58874f3692f9ca9011d84d70e8f74dad430bf8a58
c8330fb0927e6048db45e3b18544a2d780e005903a668bbfb
a9fa7f0b5522ba0719c4e643c9fea11563fbd62d594c459d8
25f96ef53ed81aab2ec6a0657288c799e418e7e2d0451ca52
```

```
35a0d7dde3fe13f31cd6e7bd46470a58c2cb7372587b8e306
84fa98c2b6ddf3e06392bcbe41b80d107790ea58cc5b52115
6264ac9051b1b4aed931f9ba5b2bbe7e12b62de036f7cd1fb
328bfe80c8781a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response [HTTPS + JSON]

HTTPS STATUS CODE : 200 OK

```json
{
  "data": {
    "uuid": " 00d8f6a8-edbe-4eee-9881-b347b740c852",
    "paymentInfoId":"e6dcb582-eec9-49ca-9faf-0e01de779447",
    "transaction": {
      "transactionOrderId":"9ef2eb6a-e265-4903-9f82-81f35ac6bc59",
      "amount": 10,
      "currencyCode": "GBP",
      "transactionId": " 201607080000584",
      "pspCode": "SCTD",
      "pspTid": "4-9-2733014",
      "region": "UK"
    }
  }
}
```

Failed Response [HTTPS + JSON]

HTTPS STATUS CODE : 500 Internal Server Error

```json
{
  "error": {
    "code": 941,
    "message": "Failed to payout payments.",
    "detail": "VPay or Maestro Card Type could not pay out"
  }
}
```

DELETE Refund Transaction (RefundTransaction)

Description

A server can invoke this action to refund a transaction.

Endpoint url

/billing/{uuid}/transactions?referenceTransactionId={referenceTransactionId}&note={note}

HTTPS Method

DELETE

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	36	M	User's unique Identification.
orderId	String	36	M	Unique OrderId for a transaction by the client side.
referenceTransactionId	String	36	M	Corresponds to a Deposit Transaction ID of Example. For example: "201607070000560".
note	String	200	O	A note.
amount	Double	8	O	The specific amount for a partial refund. If the amount is not set, it will perform a full-amount refund.

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
uuid	String	36	User's unique ID.
transaction	Array	N/A	Array of the transaction object.
transactionID	String	50	Unique ID of Example.
transactionOrderID	String	36	Unique orderId for a transaction by the client side.
referenceTransactionId	String	36	Corresponds to a Deposit Transaction ID of Example.
currencyCode	String	3	Payment currency code: ISO 4217 *(https://en.wikipedia.org/wiki/ISO_4217)*
amount	Double	8	Payment amount.
pspCode	String	10	Payment Service Provider Code "SCTD" : SecureTrading.
pspTid	String	50	Unique ID from a Payment Service Provider - "Claim by user".
region	String	40	Region for a user's payment information: • "Domestic" • "Intra Europe EEA" • "Intra Europe (Non-EEA)" • "International"

HTTPS Request example: RefundTransaction [DELETE method]

```
                                                                                https
billing/32303136303730373138343033393037303 2
3435656366636234f12bc1f26190f62a6c5565c42700f7a54bd
f3e1320badc6dce16e1d08735a5e1116c0ea02b5d127f9ad34c
88fb2852c598792709/transactions?
data=32303136303730373138343033393839393064316433 61
656435ad2e0cfc1be89bb283120eb72acf333c4839576d8e630
```

```
57c44b92ab05985e64c329279a6de4cc9f3c61b181b81273cc3
4c93ea2b266659342d026638b4ef077b08578e4468e38a6805
```

Encrypting Data and HTTPS Request [DELETE method]

```
https

DELETE
http(s)://{host}/billing/32303136303730373731383430333
9303730323435656366636234f12bc1f26190f62a6c5565c4270
0f7a54bdf3e1320badc6dce16e1d08735a5e1116c0ea02b5d127
f9ad34c88fb2852c598792709/transactions?
data=32303136303730373731383430333938393930643164334336
56435ad2e0cfc1be89bb283120eb72acf333c4839576d8e63057
c44b92ab05985e64c329279a6de4cc9f3c61b181b81273cc34c9
3ea2b266659342d026638b4ef077b08578e4468e38a6805

"Authorization: PLTOKEN {TOKEN}"
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

```
https


32303135303932343135303433323566623536346361323
438818b824cf62c2d63feba0314c8b6cd3b328c2f02380787
406779be1607fa68363bf1fbac60a1b691de2f283ee2370f0
4f479287589b712576357c3256a2ba87f9c24d5948bb91dd1
3211ed9e452247603a75b5f5672f6718242e78baf48524182
e5f35ffa58874f3692f9ca9011d84d70e8f74dad430bf8a58
c8330fb0927e6048db45e3b18544a2d780e005903a668bbfb
a9fa7f0b5522ba0719c4e643c9fea11563fbd62d594c459d8
25f96ef53ed81aab2ec6a0657288c799e418e7e2d0451ca52
35a0d7dde3fe13f31cd6e7bd46470a58c2cb7372587b8e306
84fa98c2b6ddf3e06392bcbe41b80d107790ea58cc5b52115
6264ac9051b1b4aed931f9ba5b2bbe7e12b62de036f7cd1fb
328bfe80c8781a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response

HTTPS STATUS CODE : 200 OK

```json
{
  "data": {
    "uuid": "9f3e9c0f-b7fe-4817-8bf9-f6ed55fa35be",
    "transaction": {
      "amount": 10,
      "currencyCode": "GBP",
      "transactionOrderId": "9ef2eb6a-e265-4903-9f82-81f35ac6bc59",
      "transactionId": "201607070000574",
      "referenceTransactionId": "201607070000556",
      "pspCode": "SCTD",
      "pspTid": "5-9-2491291",
      "region": "UK"
    }
  }
}
```

Failed Response [HTTPS + JSON]

HTTPS STATUS CODE : 400 Bad Request

```json
{
  "error": {
    "code": 944,
    "message": "PSP refund processing is failed.",
    "detail": "[20007] Refund amount too great"
  }
}
```

PUT Reject Transaction (RejectTransaction)

Description

A server can invoke this action to reject a transaction.

Endpoint url

/billing/{uuid}/transactions/{referenceTransactionId}/status

HTTPS Method

PUT

HTTPS Request data: Parameters

Note: **M/O** = Mandatory or Optional; **Size** = number of characters

Parameter	Type	Size	M/O	Description
uuid	String	36	M	User's unique ID.
orderId	String	36	M	Unique OrderId for a transaction by the CORE admin side.
referenceTransactionId	String	36	M	Corresponds to a Deposit Transaction ID of Example. For example: "201607070000560".
cancelReason	String	200	M	The reason for the cancellation.

HTTPS Response data: JSON Properties

JSON Property	Type	Size	Description
uuid	String	36	User's unique ID.
transaction	Array	N/A	Array of the transaction object.
transactionOrderID	String	36	Unique OrderId for a transaction by the CORE admin side.
referenceTransactionId	String	36	Corresponds to a Deposit Transaction ID of Example.
pspCode	String	10	Payment Service Provider Code "SCTD" : SecureTrading.

HTTPS Request example: RejectTransaction [PUT method]

```
https
billing/
32303136303832393231333235313261613765356239396438612
934e764f14ab49f2e028d6cbdd5b8daeb2986e28efa834833efd2
34be21e1efdf482168bb580cc3b6c98cfbfdc0b8efa401fd43/
transactions/32303136303832393231333235313164663365 36
6339663731626f99bf975a437508fe85af238214fa3768c6a1b64
4dd5092d6687583d184b1/status
```

Encrypting Data and HTTPS Request [PUT method]

```
https
PUT
http(s)://{host}/billing/32303136303730373731383430333
9303730323435656366636234f12bc1f26190f62a6c5565c4270
0f7a54bdf3e1320badc6dce16e1d08735a5e1116c0ea02b5d127
f9ad34c88fb2852c598792709/transactions?
data=32303136303730373138343033393839393064316433616
56435ad2e0cfc1be89bb283120eb72acf333c4839576d8e63057
c44b92ab05985e64c329279a6de4cc9f3c61b181b81273cc34c9
3ea2b266659342d026638b4ef077b08578e4468e38a6805

"Authorization: PLTOKEN {TOKEN}"
```

Successful *Encrypted* Response

HTTPS STATUS CODE : 200 OK

```https
32303135303932343135303433323566623536346361323133
438818b824cf62c2d63feba0314c8b6cd3b328c2f02380787
406779be1607fa68363bf1fbac60a1b691de2f283ee2370f0
4f479287589b712576357c3256a2ba87f9c24d5948bb91dd1
3211ed9e452247603a75b5f5672f6718242e78baf48524182
e5f35ffa58874f3692f9ca9011d84d70e8f74dad430bf8a58
c8330fb0927e6048db45e3b18544a2d780e005903a668bbfb
a9fa7f0b5522ba0719c4e643c9fea11563fbd62d594c459d8
25f96ef53ed81aab2ec6a0657288c799e418e7e2d0451ca52
35a0d7dde3fe13f31cd6e7bd46470a58c2cb7372587b8e306
84fa98c2b6ddf3e06392bcbe41b80d107790ea58cc5b52115
6264ac9051b1b4aed931f9ba5b2bbe7e12b62de036f7cd1fb
328bfe80c8781a33c3d63c923322a2d55fb5
```

Successful *Decrypted* Response [HTTPS + JSON]

HTTPS STATUS CODE : 200 OK

```json
{
  "data": {
    "uuid": "139f0344-0641-4d9f-a47f-e695d05842a9",
    "transaction": {
      "transactionOrderId":"ed1b1a2d-5a55-48ed-a95e-93ef732193ca",
      "referenceTransactionId": "201608290000122",
      "pspCode": "SCTD",
    }
  }
}
```

Failed Response [HTTPS + JSON]

HTTPS STATUS CODE : 400 Bad Request

```json
{
  "error": {
    "code": 947,
    "message": "Failed to reject payment.",
    "detail": Not existing payment request detail."
  }
}
```

Example
Tech Documentation

Secure Payment APIs v5.51

Released: **2011-01-10** | Sprint: 5.51
Security classification: **PUBLIC**

Protocol Rules

Example Secure Payment APIs Security: Encrypted Data

The Example Secure Payment APIs protect all HTTPS Request and Response data according to PCI and PII standards with the AES-GCM encryption *(https://tools.ietf.org/html/rfc5288)* (Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) as a Transport Layer Security (TLS) authenticated encryption operation). Please see the Protocol Rules "AES-GCM encrypted request and response data" section for these and other Example Secure Payment API security-related details.

The API is implemented as a RESTful (Representational State Transfer) API.

- If you make an API HTTPS Request using a correct URL with a HTTPS Method and any resource, the HTTPS Response will be presented as encrypted string data.
- This Restful API observes the HTTPS 1.1 Specification and all exchangeable messages are in UTF-8 encoded format.

Resource URL description

Resource URL syntax

Syntax: **{HTTPS_METHOD} https://{host}/{resourcepath}?{parameter}**		
Item	**Description**	**Example**
{HTTPS_METHOD}	One of several HTTPS Methods used in the API.	GET, PUT, POST, DELETE
https://{host}	The host address.	https://127.0.0.1
{resourcepath}	The resource path. Note: a resource is one of the API endpoints described in the "Payment API endpoints overview" in the Introduction page.	/billing/ 89396B99-5A57 -4ECA-96C9- 60ED109F8C8A /transactions
{parameter}	Parameters (optional)	?data=3230313530393 2323136313733306330 6261373832326330383 9e61251256b1bd3583b 2f02642a2e678debda2 f67a685d3e56e034f8

Request URL example:

```
                                                                    https
GET https://127.0.0.1/billing/
89396B99-5A57-4ECA-96C9-60
ED109F8C8A/transactions?
data=323031353039323231363
137333063306261373832326 33
03839e61251256b1bd3583b2f0
2642a2e678debda2f67a685d3e
56e034f8
```

Request and Response examples

Request examples

Using the HTTPS POST Method

```
https
POST https://{host}/{resourcepath}

"Content-Type: application/x-www-form-urlencoded"
"Authorization: PLTOKEN {TOKEN}"

data={Encrypted Request Data}
```

Using the HTTPS GET Method

```
https
GET https://{host}/{resourcepath}?data={Encrypted Request Data}

"Authorization: PLTOKEN {TOKEN}"
```

Response example

```
https
HTTPS Status Code XXX
{Encrypted Response Data}
```

AES-GCM encrypted request and response data

To tighten up security, all request and response data should be encrypted.

HTTPS Request data

All HTTPS Request data should have an encrypted string format. The encrypted request data has the following form:

```
https
PCNG Encrypt Function (Key, Data) = Encrypted Request Data
```

HTTPS Response data

The client should decrypt the response data in order to verify the result.

```
https
PCNG Decrypt Function (Key, Data) = Decrypted Response Data
```

Examples

GET Transactions

- URL : GET https://127.0.0.1/billing/{UUID}/transactions?data={data}
- UUID : 187b2be8-24fb-4892-b16a-7d84d1d0fe83
- data : pageNo=1&pageSize=10&fromDate=20150101&toDate=20150831

Step 1. Encrypt UUID

```
https
PCNG Encrypt Function (Key, UUID) =
"32303135303932333133033333034
66303638656231646137343 2ab97d
c062040796e4943cb9a2e31ab493f
4c10cc670de4d840469a7b771b59e
9432b48b69fe6dd19c41675e4ccda
06966216f1678838"
```

```
https
PCNG Encrypt Function (Key, Data) =
32303135303932323136313733306
33062613738323263303839e61251
256b1bd3583b2f02642a2e678debd
a2f67a685d3e56e034f8
```

Step 2. Send Request

```
https
URL : GET
https://127.0.0.1/billing/
32303135303932333130333330346 6
30363865623164613734 32ab97dc06
2040796e4943cb9a2e31ab493f4c1c
c670de4d840469a7b771b59e9432b4
8b69fe6dd19c41675e4ccda0696621
6f1678838/transactions?
data=32303135303932323136313733
33063306261373832326330 3839e61
251256b1bd3583b2f02642a2e678de
bda2f67a685d3e56e034f8
```

Step 3. Decrypt Response

Success Case

```https
32303135303932333133033333432
353433313832636138663238864934
fe9dd274394853102e05b1d5028626
dfb9f23ae17cc4f8c4060b20844036
747de4a96d7d876ca639a7e83fae26
e3e82bfa0f5d7371096a7bac8a95f5
c0c23fa363d6dbd285227c33db95f1
98f5c2f797d24d4a7f8bcc43bb9391
b0c4561a10c80f880d8f0f82b1c45b
58aa3db8f9ba
```

PCNG Decrypt Function (Key, Success Response Data)

```json
{
  "orderID": "83C2A1E7-BF50-440C-AB94-DF24E0533145",
  "applicationId": "ABC",
  "amount": 10,
  "currencyCode": "GBP",
  "ipAddr": "1.1.1.1",
  "paymentInfoID": "517c654b-e956-49e5-a0c5-4d6de609b6dd",
  "paymentInfo": {
    "paymentType": "CreditCard",
    "card": {
      "cardCVVEncKey": "94b2d1f484d3410b90b1d7d52d3479ea"
    }
  }
}
```

Failure Case

```https
32303135303932333133033363433383
865333831323532303961801 6d11daa43e01
68437b21d16cb8acbbe46ed66b125248d30c
bcf47a327097deeca14bb9004f1fb3872338
134ca82bc18875fef763ef05d92c84e76bda
4aa9bd4223b6d6f0a678f6fa32cf97e21759
59806df5dffd93dca8d6acd854ed17fd0d5a6
```

PCNG Decrypt Function (Key, Failure Response Data)

```json
{
  "error": {
    "code": 997,
    "message": "The request is invalid.",
    "detail": "Invalid uuid"
  }
}
```

Create Billing Transaction

URL : POST https://127.0.0.1/billing/{uuid}/transaction

Request Data

```json
{
  "orderID": "83C2A1E7-BF50-440C-AB94-DF24E0533145",
  "applicationId": "ABC",
  "amount": 10,
  "currencyCode": "GBP",
  "ipAddr": "1.1.1.1",
  "paymentInfoID": "517c654b-e956-49e5-a0c5-4d6de609b6dd",
  "paymentInfo": {
    "paymentType": "CreditCard",
    "card": {
      "cardCVVEncKey": "94b2d1f484d3410b90b1d7d52d3479ea"
    }
  }
}
```

Step 1. Encrypt Content

```https
PCNG Encrypt Function (Key, Request Data) =
"32303135303932333130333303466303638
6562316461373432ab97dc062040796e4943c
b9……”
```

Step 2. Send Request

URL : POST https://127.0.0.1/billing/3230313530393.../transaction
Content-Type : application/x-www-form-urlencoded
Content:

```https
data=32303135303932333130333303466
3036386562316461373432ab97dc06204079
6e4943cb9……
```

Step 3. Decrypt Response

Success Case

```https
32303135303932333130353035336343937
3632326336613936c00d50a2157b281e6360
7ef843a87d10cf05cd4669fde2b8cafa3f3c
8fc1f5f57771e6ee6f3f653d869bcd4f0151
3230f933b4a8502f00ff1a608367064b803f
af0ed02d06ecd928f18b0c15f055116c4bff
c9dd711e5eb150f7757d6132206020605d6c
85605e5e34…
```

PCNG Decrypt Function (Key, Success Response Data)

```json
{
  "data": {
    "uuid": "187b2be8-24fb-4892-b16a-7d84d1d0fe83",
    "orderID": "939da16f-a4ac-4c15-86cb-3dc9602197d4",
    "transactionID": "201509230000066",
    "amount": 10,
    "currencyCode": "GBP",
    "paymentDate": "23/09/2015 10:51:22",
    "paymentInfo": {
      "paymentType": "creditcard",
      "card": {
        "cardBrand": "Visa",
        "cardLastNum": "1881"
      }
    }
  }
}
```

Failure Case

```https
32303135303932333130333634333838
65333831323532303961801 6d11daa43e0168
437b21d16cb8acbbe46ed66b125248d30cbcf
47a327097deeca14bb9004f1fb3872338134c
a82bc18875fef763ef05d92c84e76bda4aa9b
d4223b6d6f0a678f6fa32cf97e2175959806d
f5dffd93dca8d6acd854ed17fd0d5a6
```

PCNG Decrypt Function (Key, Failure Response Data)

```json
{
  "error": {
    "code": 997,
    "message": "The request is invalid.",
    "detail": "Invalid uuid"
  }
}
```

Response data type

All HTTPS Responses will return an AES-GCM encrypted string in the following format.

```https
3230313530393232313631373330633062 6
13738323263303839e61251256b1bd3583b2f02642a
2e678debda2f67a685d3e56e034f827047c583ac13a
76dc69c498e2a92789a9bf644c3f6a1ab07859646a8
87827189e452fd1e94f3faf6061cdf42297b6b2e981
b8c9b40e5c7ac98a534439a32d32df42a68ceb19226
2a8fb3872cc4b20ad3d5aa5cbba95b8eba26e45b302
b8d796f29e37e65575ec41d1b2be67d88404fa225aa
6e999274b563f7b2c207c3
```

To verify a response, you should decrypt the response data. If the response data is successfully decrypted, you can confirm the JSON data.

```json
{
  "data": {
    "authTransID": "E37C8425-B947-4EB3-903C-8E44E916DDFD",
    "authDate": "12/11/2015 10:01:01"
  }
}
```

How to verify HTTPS Response data

If you decrypt HTTPS Response string data:

- The **success** response has a "data" object.
- The **fail** response has an "error" object.

A HTTPS Status Code of 200 or 201 means successful transactions; otherwise a different code means a failed transaction.

- In the case of a failed transaction, you should refer to the "error" object which includes a relevant error description.
- Eventually, you should double check the validation of the transaction by checking if the HTTPS Status Code is "200" and has a "data" object.

Success case : HTTPS Status Code = 200 AND a data object.

```json
200 OK
{
  "data": {
    "authTransID": "E37C8425-B947-4EB3-903C-8E44E916DDFD",
    "authDate": "12/11/2015 10:01:01"
  }
}
```

Failure case : HTTPS Status Code != 200 AND has an error object.

```json
404 Not Found
{
  "error": {
    "code": 901,
    "message": "This account (89396B99-5A57-4ECA-96C9-60ED109F8C8A) does not exist.",
    "detail" : null
  }
}
```

Security (HMAC Authentication)

A. The Authentication Header using API Key Authentication

This RESTful API uses the HTTPs Authorization header to pass authentication information. The Authorization header has the following form:

```https
Authorization : PLTOKEN {APP_ID}:{Signature}:{Nonce}:{Timestamp}
```

"PLTOKEN"
This is a predefined custom scheme by Example for an Authorization header. (Not changed.)

"APP_ID" and "APP_KEY"
These will be provided by Example. APP_ID is public. BUT, *APP_KEY* should **ONLY** be shared between Example and Example.

"Signature"
This is a hashed value produced by using the **HMAC-SHA256** algorithm.

"Nonce"
This is generated at Client and should be a unique value on each request. Example receives the request and checks to prevent a replay attack.

- Example will cache Nonce during a period of 5 minutes. Therefore, a request that has the same Nonce will be rejected during those 5 minutes.
- Nonce should be generated as a unique value.

"Timestamp"
This is calculated using a UNIX timestamp based on UTC Time to overcome different Time Zone issues between Client and Server.

- **Note:** If the time gap is *longer than 5 minutes*, your request will be failed.

B. Constructing the Signature

Note: Please keep the order of the parameters and their encoding as shown in the following diagram.

```https
Authorization = "PLTOKEN" + " " + "APP_ID" + ":" + Signature + ":" + Nonce + ":" + Timestamp

Signature = Base64 ( HMAC-SHA256 (Decode(APP_KEY) , UTF-8-Encoding ( RequestString ) ) )

RequestString = APP_ID +
                Request-HTTPS-Method +
                URI-Encoding( LowerCase ( Request-Uri) ) ) +
                UNIX-Timestamp +
                Nonce +
                Base64( MD5 ( Request-Content) ) )
```

C. HTTPS Request example

Request
GET https://127.0.0.1/account/89396B99-5A57-4ECA-96C9-60ED109F8C8A/paymentinfo

Credential **(Note: The following credential is an example. *Please do not use it.*)**
APP_ID : 44f0878d9bc947e6a4be54aec17632a5
APP_KEY : 6cuzw6qG3vsJ3DIEm+PmJe37Jm1ceOmyUIj2T18VWYHA=

Please see the following steps:

Step 1.
Generate the UUID as a Nonce.

Step 2.
Calculate the Unix Time stamp based on UTC Now.

Step 3.
Make the Signature.

Field	Value
APP_ID	44f0878d9bc947e6 a4be54aec17632a5
Request-HTTPS-Method	GET
URI-Encoding (LowerCase(Request-Uri)))	https https%3a%2f%2f127.0.0.1 %2faccount%2f9396b99-5 a57-4eca-96c9-60ed109f 8c8a%2fpaymentinfo
UNIX-Timestamp	1441009867
Nonce	27071fad-ad4d-4abe -b250-7a20679cd1e3
Base64 (MD5(Request-Content)))	-

Make "RequestString" concatenate values:

```
https
RequestString =
44f0878d9bc947e6a4be54aec17632a5GEThttp
%3a%2f%2f127.0.0.1%2faccount%29396b99-5
a57-4eca-96c9-60ed109f8c8a%2fpaymentinfo
144100986727071fad-ad4d-4abe-b250-7a2067
9cd1e3
```

Make a "Signature" hashing of the above RequestString using APP_KEY:

```
https
Signature = 0/V3xVN8rzfU8fj6lLTSIHF
co9NnXBXVGHlKIkiwKUc=
```

Step 4.
Add the "Authorization Header" to the Request

```
https
Authorization: PLTOKEN
44f0878d9bc947e6a4be54aec17632a5:0/V3x
VN8rzfU8fj6lLTSIHFco9NnXBXVGHlKIkiwKUc
=:27071fad-ad4d-4abe-b250-7a20679cd1e3
:1441009867
```

Error Handling (HTTPS Status Codes and Error object)

When the API encounters any errors, the corresponding **HTTPS Status Code** and **Error Object** will be returned in the following JSON format:

```json
HTTPS Status Code XXX
  {
    error{
      "code": (Error code),
      "message": "(Error description)",
      "details": "(Additional details about the error)"
    }
}
```

Common Error Code list

Code	Message Name	Message	HTTPS Status Code
901	ACCOUNT_NOTFOUND_ERR	This account does not exist. Cannot find related account by requested uuid.	**404: Not Found**
903	PAYMENTINFO_NOTFOUND_ERR	This account's payment information does not exist. Cannot find related payment method by requested uuid and pmid.	**404: Not Found**
911	TRANSACTION_NOTFOUND_ERR	This account does not have transactions. Cannot find related transactions.	**404: Not Found**
912	REQUEST_NOTFOUND_ERR	Request does not exist. Cannot find related request log from mobile client.	**404: Not Found**
932	PAYMENTINFO_VERIFICATION_PSP_ERR	PSP verification processing failed. Detailed information about the reason of failure (PSP error etc.). Payment Service Provider returns error when verifying the payment method. "detail" attribute describes the reason of failure.	**400: Bad Request**
933	PAYMENTINFO_CREATE_ERR	Failed to create payment information. Error occurred in Example process while user creates a payment method. "detail" attribute describes the reason of failure.	**400: Bad Request**
934	PAYMENTINFO_UPDATE_ERR	Failed to update payment information. Error occurred in the Example process while user updates payment method. "detail" attribute describes the reason of failure.	**400: Bad Request**
935	PAYMENTINFO_DELETE_ERR	Failed to delete payment information. Error occurred in the Example process while user deletes payment method.	**400: Bad Request**
936	PAYMENT_CREATE_ERR	Failed to create payments. Error occurred in Example process while user creates payment transaction.	**400: Bad Request**
937	PAYMENTINFO_REVERIFY_ERR	Failed to re-verify payment information. Error occurred in Example process while user re-verifies payment method.	**400: Bad Request**

Code	Message Name	Message	HTTPS Status Code
938	PAYMENT_CREATE_PSP_FAIL_ERR	PSP payment processing failed. Payment Service Provider returns error when authorizing payment transaction. Detailed information about the reason of failure (PSP error etc.). "detail" attribute describes reason of failure.	**400: Bad Request**
939	PAYMENT_CHECK_ERR	Failed to check to create payments. Error occurred in Example process while checking 3DS and Sphonic.	**400: Bad Request**
940	PAYMENT_PSP_ROUTER_ERR	There is no available Payment Service Provider. Please try again. Error is occurred in the Example process while routing PSP.	**400: Bad Request**
941	PAYMENT_PAYOUT_ERR	Failed to payout payments. Error occurred in Example process while CORE makes payout transaction.	**400: Bad Request**
942	PAYMENT_PAYOUT_PSP_ERR	PSP payout processing is failed. Error occurred in Example process while CORE makes payout transaction. "detail" attribute describes reason of failure.	**400: Bad Request**
943	PAYMENT_REFUND_ERR	Failed to refund payments Error occurred in Example process while CORE makes refund transaction.	**400: Bad Request**
944	PAYMENT_REFUND_PSP_ERR	PSP refund processing is failed. Detailed information about the reason of failure (PSP error etc.) is provided. Error occurred in a Example process while CORE makes refund transaction. "detail" attribute describes reason of failure.	**400: Bad Request**
945	PAYMENT_3DS_PSP_ERR	PSP 3DS processing is failed. Detailed information about the reason of failure (PSP error etc.) is provided. Payment Service Provider returns error when proceeding with the 3DS process. "detail" attribute describes reason of failure.	**400: Bad Request**
946	PAYMENT_PSP_TIMEOUT	PSP Timeout. Connection with the Payment Service Provider has timed out.	**400: Bad Request**
947	PAYMENT_REJECT_ERR	Failed to reject payment. Error occurred in the Example process while CORE makes a reject transaction.	**400: Bad Request**
949	PAYMENTINFO_CANCEL_ERR	Request is canceled by the mobile client. Mobile client made a request using a canceled orderId.	**400: Bad Request**
950	PAYMENT_CANCEL_ERR	Request is canceled by the mobile client. Mobile client made a request using a canceled orderId.	**400: Bad Request**
974	PAYMENTINFO_CREATE_FAIL_ERR	An unexpected error occurred during create account payment information. Please try again. Unexpected error occurred in Example process while user creates payment method.	**500: Internal Server Error**

Code	Message Name	Message	HTTPS Status Code
975	PAYMENTINFO_UPDATE_FAIL_ERR	An unexpected error occurred during update account payment information. Please try again. Unexpected error occurred in Example process while user updates payment method.	**500: Internal Server Error**
976	PAYMENTINFO_DELETE_FAIL_ERR	An unexpected error occurred during delete account payment information. Please try again. Unexpected error occurred in Example process while user deletes payment method.	**500: Internal Server Error**
979	PAYMENTINFO_REVERIFY_FAIL_ERR	An unexpected error occurred during re-verify account payment information. Please try again. Unexpected error occurred in Example process while user reverifies payment method.	**500: Internal Server Error**
981	PAYMENT_CHECK_FAIL_ERR	An unexpected error occurred during check create payment. Please try again. Unexpected error occurred in Example process while checking 3DS and Sphonic.	**500: Internal Server Error**
982	PAYMENT_CREATE_FAIL_ERR	An unexpected error occurred during create payment. Please try again. Unexpected error occurred in Example process while user creates payment transaction.	**500: Internal Server Error**
984	PAYOUT_CREATE_FAIL_ERR	An unexpected error occurred during payout payment. Please try again. Unexpected error occurred in Example process while CORE makes payout transaction.	**500: Internal Server Error**
985	PAYMENT_REFUND_FAIL_ERR	An unexpected error occurred during refund payment. Please try again. Unexpected error occurred in Example process while CORE makes refund transaction.	**500: Internal Server Error**
987	PAYMENT_REJECT_FAIL_ERR	An unexpected error occurred during a reject payment. Please try again. Unexpected error occurred in a Example process while CORE makes a reject transaction.	**500: Internal Server Error**
993	AUTHORIZATION_ERR	You do not have authorization. The client does not have an authorization target in the API.	**403: Forbidden**
994	NOTIMPLEMENTED_METHOD_ERR	Related function is not implemented. The client invokes no implemented API.	**501: Not Implemented**
995	NOTALLOWED_METHOD_ERR	Requested method is not allowed. The client invokes a non-existing method.	**405: Method Not Allowed**
996	RESOURCE_NOTFOUND_ERR	Cannot find resource. Please check resource. The client invokes a non-existing resource.	**404: Not Found**
997	REQUEST_ERR	The request is invalid. Information is provided about the invalid field. Client doesn't send to Example a required attribute. "detail" attribute describes reason of failure.	**400: Bad Request**
998	AUTHENTICATION_ERR	Authentication token is missing or incorrect. The client does not have authentication.	**401: Unauthorized**
999	API_INTERNAL_ERR	Internal Error. Unexpected error occurred. "detail" attribute describes reason of failure.	**500: Internal Server Error**

Code	Message Name	Message	HTTPS Status Code
1001	CORE_API_SEND_ERR	CORE Service Error (detailed CORE Service Error). CORE returns error. "detail" attribute describes reason of failure.	**400: Bad Request**
1600	SAKA_GET_CVV_ERR	SAKA error - Get CVV Cannot get CVV from SAKA.	**400: Bad Request**
1601	SAKA_GET_CARD_DETAIL_ERR	SAKA error - Get Card Detail Cannot get Card Details from SAKA.	**400: Bad Request**
1602	SAKA_ENC_CARD_DETAIL_ERR	SAKA error - Tokenized Card Detail Cannot tokenize Card Details with SAKA.	**400: Bad Request**
1603	SAKA_ENC_CARD_SUMMARY_ERR	SAKA error - Tokenized Card Summary Cannot tokenize Card Summary with SAKA.	**400: Bad Request**
1604	SAKA_ENC_CARD_NUM_ERR	SAKA error - Tokenize Card Number Cannot tokenize Card Number (PAN) with SAKA.	**400: Bad Request**
1700	SPHONIC_CHECK_ERR	Sphonic error - Cannot check Risk level. If Sphonic does not return anything, or failed to connect to Sphonic. "detail" attribute describes reason of failure.	**400: Bad Request**
1701	SPHONIC_RISK_TRANSACTION_ERR	Sphonic error This transaction is detected as risk transaction. Detailed Sphonic Error. Occurs ff Sphonic returns "Fraud" while checking AML. "detail" attribute describes reason of failure.	**400: Bad Request**

Example
Tech Documentation

Secure Payment APIs v5.51

Released: **2011-01-10** | Sprint: 5.51
Security classification: **PUBLIC**

3DSecure integration

Example Secure Payment APIs Security: Encrypted Data

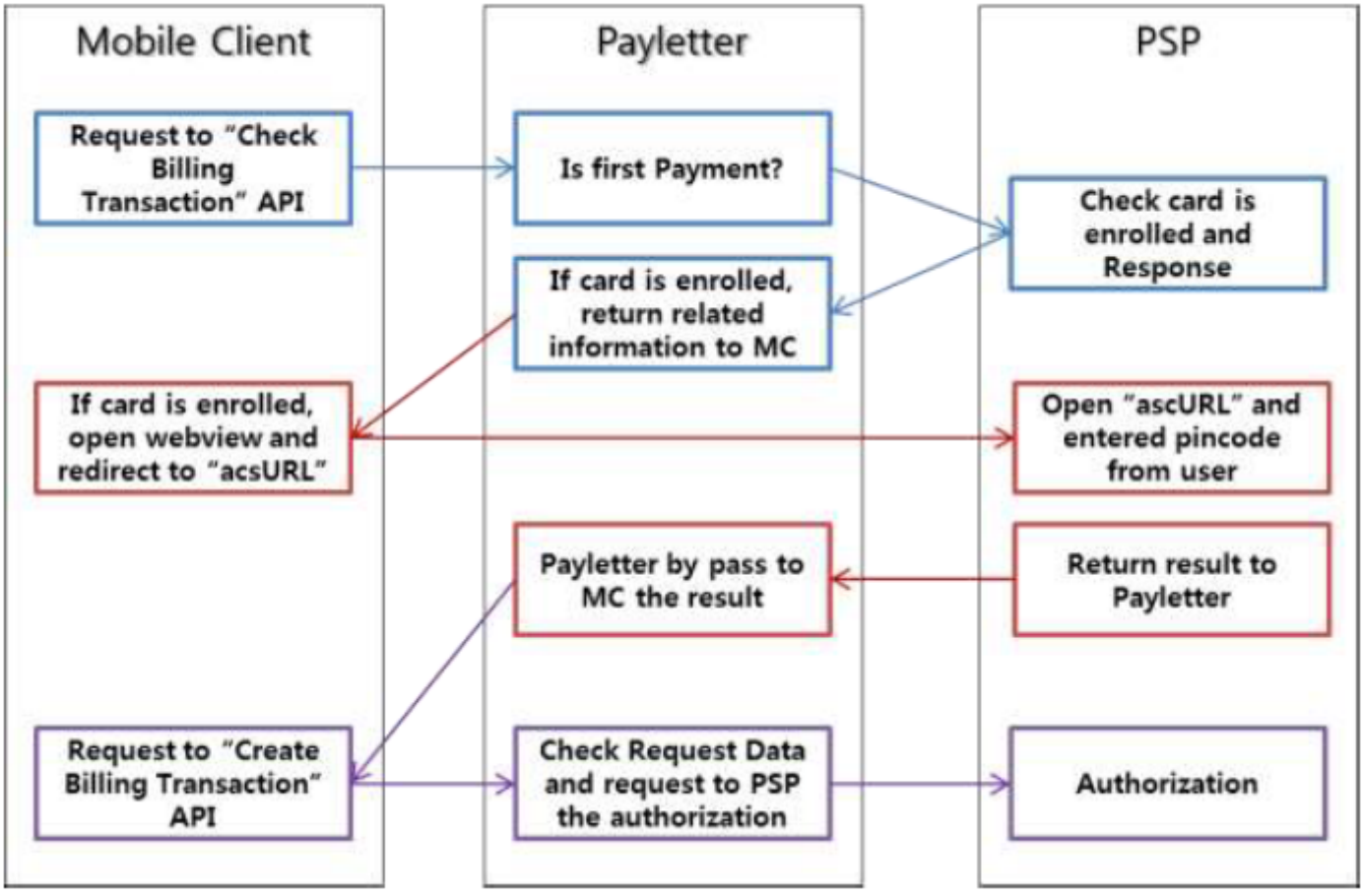
The Example Secure Payment APIs protect all HTTPS Request and Response data according to PCI and PII standards with the AES-GCM encryption *(https://tools.ietf.org/html/rfc5288)* (Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) as a Transport Layer Security (TLS) authenticated encryption operation). Please see the Protocol Rules "AES-GCM encrypted request and response data" section for these and other Example Secure Payment API security-related details.

Mobile Client URL Scheme for 3DSecure

Example Development Server : **Example://validation-complete**

3DSecure Workflow

The following diagram illustrates the 3DSecure workflow process.



Enabling 3DSecure for Secure Trading

If 3DSecure is enrolled, a Mobile client should open a web view with the "Check Billing Transaction" API response. For example:

```html
<!DOCTYPE html PUBLIC '-//W3C//DTD HTML 4.01//EN'>
  <html>
    <head>
      <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
      <title>Processing payment</title>
      <style type="text/css">
        h3, h3, h4 {
          font-family: verdana, arial, sans-serif;
          font-weight: normal;
        }
        #logo {
          float: left;
        }
      </style>
    </head>
    <body OnLoad="document.form.submit();">
      <form name="form" id="form" action="ascURL " method="POST">
      <div>
        <input type="hidden" name="PaReq" value="paReq " />
        <input type="hidden" name="TermUrl" value="termURL " />
        <input type="hidden" name="MD" value="md " />
      </div>
      <noscript>
      <div>
        <h3>JavaScript is currently disabled or is not supported by your browser.</h3>
          <h4>Please click Submit to continue processing your 3-D Secure transaction.</h4>
            <input type="submit" value="Submit">
      </div>
      </noscript>
      </form>
  </body>
</html>
```

If the 3DSecure process succeeds, Secure Trading will send the result to a Example URL. Example will then pass the result to the mobile client (3.2. Mobile Client URL Scheme for 3D Secure - Example Development Server : Example://validation-complete) in order to request the "Create Billing Transaction" APIs. For example:

```http
Example://validation-
complete?md=UEZOVVBqeE5SRDQ4VFVSSVBsWT…SmdDWW89&pares=eJztWMmyo8a2nfMVDnuoK……..vz7FfH2meb0kv5
65n29n3z5%2f%2fwv6Em60%0d%0a
```

Please note that all parameters in the preceding example are encoded for special characters.

Example
Tech Documentation

Secure Payment APIs v5.51

Released: **2011-01-10** | Sprint: 5.51
Security classification: **PUBLIC**

Testing with 3DSecure

For further 3DSecure test details and information, please refer to the Secure Trading documentation available at their website: **STPP Testing (PDF)** *(http://www.securetrading.com/files/documentation/STPP-Testing.pdf)*

Example Secure Payment APIs Security: Encrypted Data

The Example Secure Payment APIs protect all HTTPS Request and Response data according to PCI and PII standards with the AES-GCM encryption *(https://tools.ietf.org/html/rfc5288)* (Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) as a Transport Layer Security (TLS) authenticated encryption operation). Please see the Protocol Rules "AES-GCM encrypted request and response data" section for these and other Example Secure Payment API security-related details.

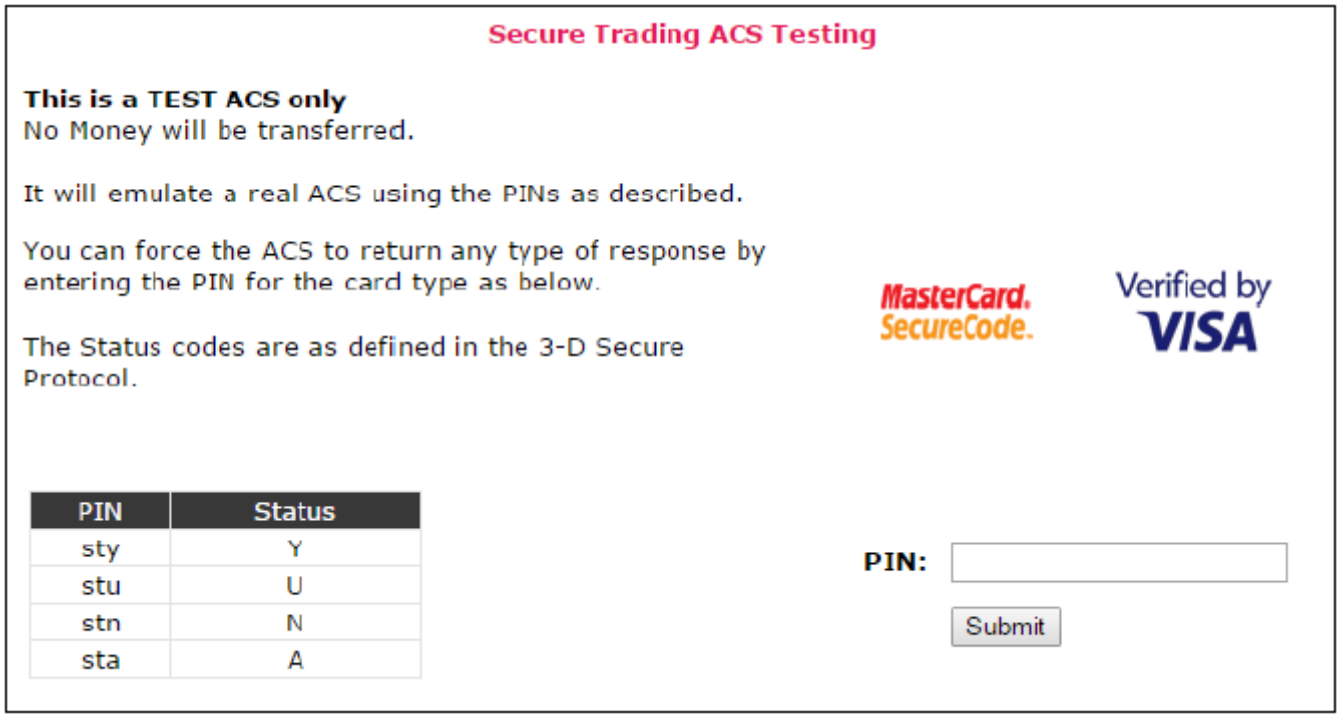
Test credit card numbers that include 3DSecure

Use these to simulate authorizations with cards that are either enrolled or not enrolled in the card issuer's 3DSecure scheme:

Card Type	Enrolled "Y"	Enrolled "N"	Enrolled "U"
Maestro	5000 0000 0000 0611	5000 0000 0000 0421	5000 0000 0000 0801
Maestro	6759 0000 0000 0711	6759 0000 0000 0521	6759 0000 0000 0901
MasterCard	5100 0000 0000 0511	5100 0000 0000 0321	5100 0000 0000 0701
MasterCardDebit	5124 9900 0000 0911	5124 9900 0000 0721	5124 0000 0000 0101
V PAY	4370 0000 0000 0111	4370 0000 0000 0921	4370 0000 0000 2307
Visa	4111 1100 0000 0211	4111 1100 0000 0021	4111 1100 0000 0401
Delta	4006 2600 0000 2473	4006 2600 0000 2481	4006 2600 0000 2408
Electron	4245 1900 0000 0311	4245 1900 0000 0121	4245 1900 0000 0501
Purchasing	4484 0000 0000 0411	4484 0000 0000 0221	4484 0000 0000 0601

Enrolled "Y":

- If you test with the card number, you should check 3DSecure.
- It means you will redirect to the ASC page (Access Control Server). See the following image.
- You can enter the following PIN codes in the ACS page:
    - **stn:** If you type this PIN code and the status comes back as a "N", then the transaction will be failed.
    - **sty, stu, sta:** Using these PIN codes the transaction will be a success.



Enrolled "N":

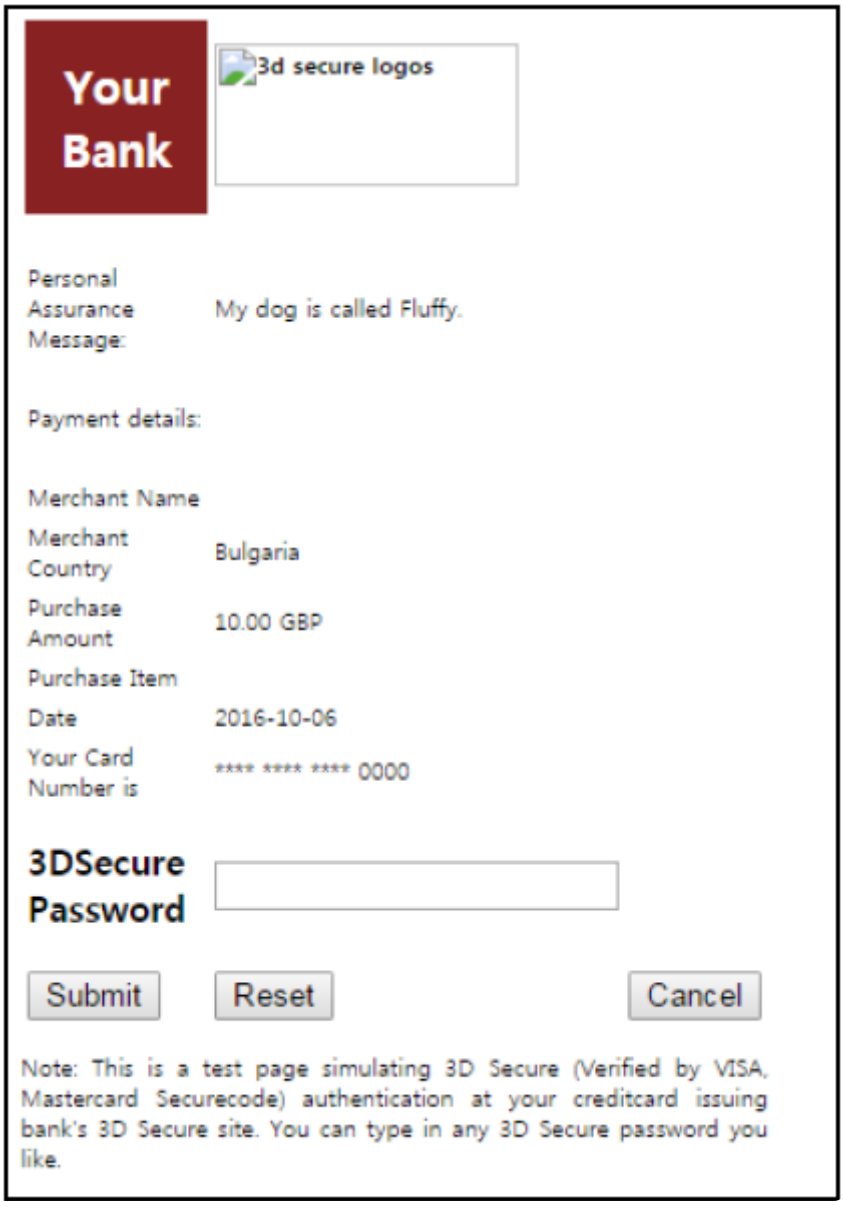
- If you test with the card number, you don't have to check 3DSecure.
- It does not redirect to the ACS Page.
- The transaction with this card number will be a success.

Enrolled "U":

- You will get a failure result.

E-Com Processing ASC Testing Page

In the E-Com Processing ACS page, you can type any 3DSecure Password. See the following image of the ACS page and section related to it below.


Your Bank
3d secure logos
Personal Assurance Message:
My dog is called Fluffy.
Payment details:
Merchant Name
Merchant Country
Bulgaria
Purchase Amount
10.00 GBP
Purchase Item
Date
2016-10-06
Your Card Number is
**** **** **** 0000
3DSecure Password
Submit
Reset
Cancel
Note: This is a test page simulating 3D Secure (Verified by VISA, Mastercard Securecode) authentication at your creditcard issuing bank's 3D Secure site. You can type in any 3D Secure password you like.

CVV Value testing

CVV2 Value	Result
123 or 333	**Success**
214 or the other numbers	**Fail**

Expiration Date testing

MM/YYYY	Result
Any date in the future	**Success**
Any date that is before the current date	**Fail**

Street Value testing

For the appropriate test, the street value should be 3 digits.

Street number	Example Street Value	Result
789 or 333	789 Main; 333 Main	**Success**
123 or the other numbers	78954 Main; 123 Main	**Fail**

Postal Code value testing

Locale Postal Code	Example Postal Code	Result
UK postal code	TE45 6ST or TE456ST or TE456STD TE33 3ST or TE333ST or TE333STQ	**Success**
US postal code	55555 or 33333	**Success**
UK postal code	TE12 3ST or TE123ST or TE12 3STD	**Fail**
US postal code	12345	**Fail**

Deposit Amount testing

Deposit Amount	Result	Description
700	**Fail**	Transaction declined by bank response.
600.10	**Fail**	Bank system error response.
The other numbers	**Success**	-

E-Com Processing Credit Card Information

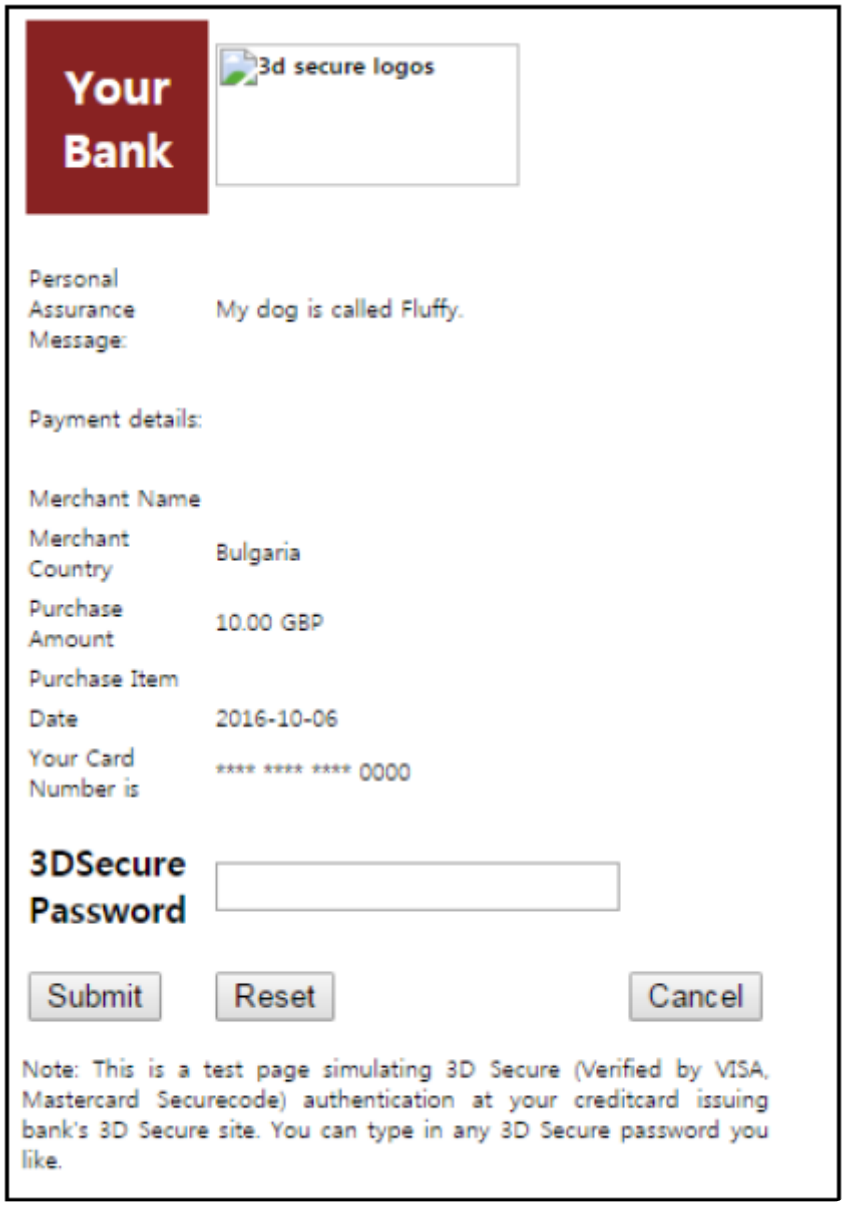
Test Credit Card numbers that include 3-D Secure.
Use these to simulate authorizations with cards that are either enrolled or not enrolled in the card issuer's 3-D Secure scheme:

Card Type	Enrolled "Y"	Enrolled "N"	Enrolled "U"

Visa	4711 1000 0000 0000	-	-
Visa	4012 0010 3685 3337	-	-

Important: Please NOTE that E-Com does not support Enrolled='N' card and MasterCard as well. They said we can test using real card on live mode.

Enrolled "Y": If you test with the card number, you should check 3DSecure. It means you will redirect to the ASC Page (Access Control Server Page) and you should enter the PIN in the ACS Page. See the following image of the ASC page.



In the E-Com Processing ACS page, you can type any 3DSecure Password.

CVV Value testing

CVV Value	Result
000 - 999	**Success**
The other numbers.	**Fail**

Expiration Date testing

MM/YYYY	Result
Any date in the future.	**Success**
Any date that is before the current date.	**Fail**

Card Holder testing

Note: The Card Holder name should be set as First Name, Last Name.

Card Holder Value	Result
Sally Kimberston	**Success**
Tester	**Fail**